UNIVERSIDADE FEDERAL DO PARANÁ

SOFIA – *SOFTWARE* OTIMIZADO DE FILTRO(S) COM INTELIGÊNCIA ARTIFICIAL

CURITIBA

DEZEMBRO/2009

ALINE ARIANE SCHULTZ
ANGÉLICA INAJÁ JULIANI
JULIANA HELENA TIBÃES
LETÍCIA FERNANDES DE JESUS
VANELY DE SOUZA

SOFIA – *SOFTWARE* OTIMIZADO DE FILTRO(S) COM INTELIGÊNCIA ARTIFICIAL

Pesquisa apresentada à disciplina de Trabalho de Conclusão de Curso, turno diurno, do Curso de Tecnologia em Sistemas de Informação da Universidade Federal do Paraná - UFPR, como obtenção de nota parcial para conclusão de curso.
Orientador: Dr. Roberto Tadeu Raittz.
Co-orientador: Michelly Alves Coutinho Gehlen.

CURITIBA
DEZEMBRO/2009

ALINE ARIANE SCHULTZ
ANGÉLICA INAJÁ JULIANI
JULIANA HELENA TIBÃES
LETÍCIA FERNANDES DE JESUS
VANELY DE SOUZA

SOFIA – *SOFTWARE* OTIMIZADO DE FILTRO(S) COM INTELIGÊNCIA ARTIFICIAL

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação, da Universidade Federal do Paraná, como requisito a obtenção do título de Tecnólogo.

COMISSÃO EXAMINADORA

_____

Prof. Dr. Roberto Tadeu Raittz
Universidade Federal do Paraná

_____

Prof. Dr. Lucas Ferrari de Oliveira
Universidade Federal do Paraná

_____

Leandro Henrique Stein (mestrando)
Universidade Federal do Paraná

Curitiba, 8 de dezembro de 2009

AGRADECIMENTOS

Primeiramente agradecemos a Deus por nos ter dado saúde e disposição para elaborar este trabalho. Às nossas famílias e namorados pela compreensão, auxílio nos momentos precisos, paciência, afeto e carinho. A essa Universidade pelas oportunidades que nos concedeu para auxiliar em nossa formação. Aos professores Rafaela e Jaime pelos conselhos no decorrer do curso e projeto final e em especial ao professor Dr. Roberto Tadeu Raittz pelo desafio dado. Aos mestrandos Michelly e Eduardo por toda ajuda no decorrer do projeto. A doutoranda Marina K. Kadowaki, ao professor Luciano Huergo e ao graduando Lucas Falarz do departamento de Bioquímica da UFPR por toda a paciência, explicação, ajuda e dicas para o sucesso do projeto. Ao Ricardo, pela ajuda no decorrer do projeto, e aos amigos pela paciência, compreensão e companheirismo nos momentos bons e ruins.

DEDICATÓRIA

“Dedico este trabalho ao meu namorado Ricardo por todo apoio, carinho e dedicação. E por sempre acreditar e torcer pelo meu sucesso.”

(Aline Ariane Schultz)

“A minha família por acreditar no meu potencial em concluir o curso e me apoiar sempre. Ao meu namorado Igor por sempre estar ao meu lado e não deixar que em momento algum eu desistisse ou duvidasse da minha capacidade.”

(Angélica Inajá Juliani)

“Ao meu irmão e ao meu namorado Mateus pela paciência, compreensão e apoio ao longo desses anos.”

(Juliana Helena Tibães)

“Dedico este trabalho à minha família e ao meu namorado, pelo apoio e paciência e por acreditarem no meu potencial.”

(Letícia Fernandes de Jesus)

“A minha família pelo esforço, dedicação, carinho e compreensão em todos os momentos desta caminhada. Aos meus amigos que compartilharam comigo alegrias e tristezas sempre me apoiando. Ao meu namorado Marcelo por estar sempre do meu lado.”

(Vanely de Souza)

RESUMO

O projeto desenvolvido visa à construção de um *software* capaz de efetuar o pré-processamento das imagens digitais de géis 2D resultantes das pesquisas proteômicas. Utiliza técnicas de inteligência artificial para a implantação de conceitos de Visão Computacional. A meta é diminuir os ruídos presentes nas imagens resultantes da corrida do gel, facilitando a detecção dos *spots* pelas soluções computacionais existentes. O modelo faz uso da linguagem de programação JAVA e suas bibliotecas padrão. Com uso de Algoritmo Genético, técnica de inteligência artificial a imagem poderá ser processada de maneira "inteligente" o que permite detectar os *spots* e eliminar os ruídos presentes. O teste da aplicação será realizado através de comparações dos resultados apresentados pelo *software* utilizado pelos pesquisadores da UFPR. Essa comparação é estabelecida com imagens submetidas e não submetidas ao pré-processamento proposto pela ferramenta desenvolvida nesse projeto.

Palavras-chave: géis 2D, inteligência artificial, Algoritmo Genético, imagens, pré-processamento.

ABSTRACT

The presented project aims the construction of a *software* capable of executing a 2D gel digital image's pre-processing resulting from proteomics research. It uses artificial intelligence techniques for the implementation of Computational View concepts. The objective is to reduce the image noise that results from the gel's run, making the detection of *spots* by the existing computational solutions easier. The model makes use of JAVA programming language and its standard libraries. With the usage of Genetic Algorithms, Artificial Intelligence technique, the image may be processed in an "intelligent" form which permits the *spots* detection and eliminate the existing image noise. The application test will be executed through the comparison of results presented by the *software* already used by the UFPR researchers. This comparison is established with images submitted and non-submitted to the pre-processing proposed by the tool developed in this project.

Key-words: 2D gel, artificial intelligence, Genetic Algorithms, images, pre-processing

LISTA DE ABREVIATURAS E SIGLAS

2D – 2 dimensões

3D – 3 dimensões

AG – Algoritmos Genéticos

API - *Application Programming Interface*

CMM – *Capability Maturity Model*

CMMI – *Capability Maturity Model Integration*

EVINCI – Evento de Iniciação Científica

GB – *Gigabyte*

HD - *Hard Disk*

IA – Inteligência Artificial

IBM – *International Business Machines*

IDE – *Integrated Development Environment*

IV & V – *Integration Validation & Verification*

JGAP – JAVA Genetic Algorithms Package

JPEG - *Joint Photographic Experts Group*

NATO – *North Atlantic Treaty Organization*

PGP – Plano de Gerenciamento de Projetos

pH – Potencial Hidrogeniônico

PMBOK – *Project Management Body of Knowledge*

RGB – *Red, Green, Blue*

SEI – *Software Engineering Institute*

SOFIA – *Software* Otimizado de Filtro(s) com Inteligência Artificial.

SQA – *Software Quality Assurance*

TIFF - *Tagged Image File Format*

UC – *Use Case*

UFPR – Universidade Federal do Paraná

UML – *Unified Modeling Language*

WBS – *Work Breakdown Structure*

XP – *Extreming Programming*

LISTA DE FIGURAS

Figura 1 - Seleção natural....16
Figura 2 - Componente do Algoritmo Genético....17
Figura 3 - Operações básicas do Algoritmo Genético....18
Figura 4 - Gel 2D....20
Figura 5 – Cromossomo com seus *genes* em azul....20
Figura 6 – Corte com sinalização dos valores do cromossomo....21
Figura 7 – Representação não exata dos cortes da imagem....21
Figura 8 – Diagrama que representa a função responsável pela nota. 22
Figura 9 - Função *Fitness*....24
Figura 10 - Imagem original....25
Figura 11 - Imagem com RGBs maiores que 145 apagados....25
Figura 12 - Imagem com RGBs maiores que 200 apagados....25
Figura 13 - Tabela usada na definição da probabilidade de ocorrência 45
Figura 14 - Tabela de impacto....45
Figura 15 - Probabilidade X Impacto 46
Figura 16 - Análise quantitativa 47
Figura 17 - WBS 54
Figura 18 - PERT/CPM....55
Figura 19 - Declaração de escopo....57
Figura 20 - Atividades....58
Figura 21 - Recursos X Cronograma....59
Figura 22 - Gráfico de *Gantt*....65
Figura 23 - Diagrama de classes....110
Figura 24 - Casos de uso....112
Figura 25 – Abertura do sistema....114
Figura 26 – Selecionar imagem....115
Figura 27 – Carregar imagem....115
Figura 28 - Carregar imagem....117
Figura 29 - Processar imagem....117
Figura 30 - Processar imagem....119
Figura 31 - Selecionar s*pot*....119

Figura 32 - Processar imagem.....121
Figura 33 - Selecionar área do gráfico.....121
Figura 34 – Gráfico.....122
Figura 35 - Seqüência - Abrir imagem.....124
Figura 36 - Seqüência - Processar imagem.....125
Figura 37 - Seqüência - Salvar imagem.....126
Figura 38 - Seqüência - Visualizar gráficos.....127
Figura 39 - Colaboração - Abrir imagem.....129
Figura 40 - Colaboração - Processar imagem.....131
Figura 41 - Colaboração - Salvar imagem.....132
Figura 42 - Colaboração - Visualizar gráficos.....133
Figura 43 - Atividades - Carregar imagem.....135
Figura 44 - Atividades - Processar imagem.....136
Figura 45 - Atividades - Selecionar s*pots*.....137
Figura 46 - Atividades - Visualizar gráficos.....138
Figura 47 - Diagrama de pacotes.....140
Figura 48 - Diagrama de componentes.....142

SUMÁRIO

1 INTRODUÇÃO 13
2 ESTUDO DE CASO 14
3 OBJETIVOS 15
3.1 OBJETIVO GERAL 15
3.2 OBJETIVOS ESPECÍFICOS 15
4 EMBASAMENTO TEÓRICO 16
4.1 ALGORITMOS GENÉTICOS 16
4.1.1 Processo de evolução natural 16
4.1.2 Componentes de um algoritmo genético 17
4.1.3 Principais conceitos 17
4.1.4 Operações básicas de um algoritmo genético 18
4.1.5 Codificação do problema para AG 19
5 METODOLOGIA 20
5.1 PLANEJAMENTO DO CROMOSSOMO 20
5.2 FUNÇÃO "*FITNESS*" 22
5.2.1 Configurando a população 26
6 PLANEJAMENTO DO PROJETO 27
7 CONTROLE DE QUALIDADE 28
8 MODELAGEM DO SISTEMA 29
9 INTERFACE 30
10 RESULTADOS 32
11 CONCLUSÃO 33
12 PROJETOS FUTUROS 34
13 BIBLIOGRAFIA 35
APÊNDICE 1 – PLANO DE GERENCIAMENTO DE PROJETOS (PGP) 36
APÊNDICE 2 – PLANO DE SQA 67

APÊNDICE 3 – DIAGRAMAS DE CLASSES.....109
APÊNDICE 4 – DIAGRAMAS DE CASO DE USO 111
APÊNDICE 5 – DIAGRAMAS DE SEQÜÊNCIA.....123
APÊNDICE 6 – DIAGRAMA DE COLABORAÇÃO.....128
APÊNDICE 7 – DIAGRAMA DE ATIVIDADES 134
APÊNDICE 8 – DIAGRAMA DE PACOTES 139
APÊNDICE 9 – DIAGRAMA DE COMPONENTES 141
APÊNDICE 10 – RESULTADO EVINCI.....143

1 INTRODUÇÃO

A presente pesquisa tem por objetivo apresentar o desenvolvimento do *software* SOFIA, englobando o planejamento, desenvolvimento das técnicas, estudos e documentação de todo seu processo.

O *software* busca por meio de uma interface intuitiva, auxiliar nas pesquisas proteômicas (semelhantes à realizada com o *Azospirillum brasiliense* do Departamento de Bioquímica da Universidade Federal do Paraná - UFPR) efetuando o pré-processamento das imagens digitais de géis 2D, resultantes do processo de eletroforese, eliminando os ruídos presentes nas imagens utilizando Algoritmos Genéticos que otimizam o processo em relação à lógica convencional.

No decorrer do projeto apresentamos uma solução para aliar a informática com a área biológica, assim tendo a Bioinformática como via evolutiva natural para ambas as áreas.

2 ESTUDO DE CASO

As proteínas são dotadas de relevantes informações biológicas, as quais são usadas para pesquisas e também descrições de sistemas biológicos. Normalmente, a separação das proteínas é feita por eletroforese bidimensional e a identificação através de especaso de testerometria de massa (JOCELYN & ILAG, 2002). A Proteômica tem como objetivo estudar as propriedades das proteínas, seus níveis de expressão, suas funções, modificações, interações entre proteínas e mecanismos regulatórios (BLACKSTOCK & WEIR, 1999).]

Segundo LOPES (1984), a eletroforese pode ser definida como a migração de modo diferencial e próprio de partículas eletricamente carregadas, quando submetida a um determinado potencial elétrico em um pH dado. Utilizando-se desta técnica, os pesquisadores do departamento de Bioquímica da UFPR, analisam os géis 2D que são proteínas de uma única amostra espalhados em duas dimensões, que consiste na primeira dimensão ser de acordo com o ponto isoelétrico (pH no qual a carga total é igual a zero) e a segunda pelo peso molecular.

O gel resultante do processo de eletroforese 2D é analisado por soluções computacionais. O *software* em questão é o *ImageMaster*^{TM} *2D Platinum* onde a imagem do gel é capturada a partir de um scanner e carregada pelo programa, que realiza o processo de análise.

Mesmo tendo disponível o *software* para análise os pesquisadores desperdiçavam muito tempo para confirmação visual do que era ou não um *spot* (ponto no gel 2D que representa uma proteína), esse problema é encontrado devido aos rastros (o que não é um *spot*) resultantes da corrida do gel que ocasionam ruídos significativos na imagem.

A proposta dessa solução computacional seria permitir o pesquisador centrar seus esforços somente na compreensão biológica do fato estudado. Desse modo o pré-processamento das imagens do gel 2D fica a cargo do *software* a ser desenvolvido.

3 OBJETIVOS

3.1 OBJETIVO GERAL

Criação de um sistema computacional capaz de minimizar os ruídos presentes em imagens de géis de eletroforese 2D, otimizando o processo de detecção de *spots* realizado pelas soluções computacionais existentes.

3.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um *software* livre, *open source*, de interface intuitiva e multiplataforma.
- Implementar conceitos de inteligência artificial através de Algoritmos Genéticos e técnicas de Visão Computacional.
- Eliminar os ruídos presentes nas imagens de géis de eletroforese 2D.
- Viabilizar o acesso às imagens processadas aos *software*s computacionais presentes no mercado.
- Desenvolver o projeto conforme o Plano Geral de Projetos. (apêndice 1)
- Desenvolver o projeto conforme o Plano de SQA. (apêndice 2)
- Desenvolver o projeto conforme a análise de UML. (apêndice 3, 4, 5, 6, 7, 8, 9)

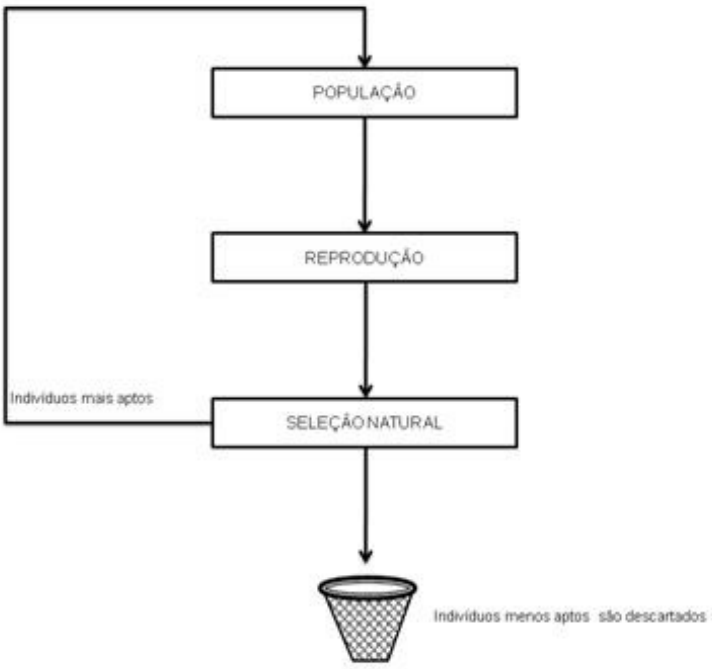
4 EMBASAMENTO TEÓRICO

Quando se trata de encontrar resultados de forma otimizada nos deparamos com pontos centrais como a definição do problema, o objetivo que deve ser atingido e o espaço de soluções que podemos apresentar. A seguir iremos apresentar a solução computacional desenvolvida para essa pesquisa.

4.1 ALGORITMOS GENÉTICOS

Os Algoritmos Genéticos são baseados na teoria da evolução de Charles Darwin (1858) por se aplicarem de forma abrangente á várias situações, simplificando a solução dos problemas que encontramos. As técnicas de AG surgiram com Jonh Holland em 1975 na Universidade de Michigan, que os define como modelos computacionais que imitam os mecanismos da “evolução natural” para resolver problemas de otimização.

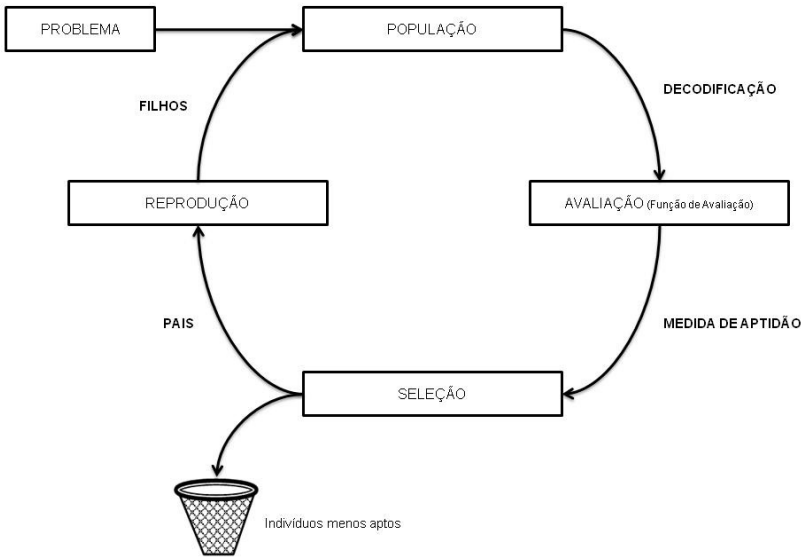
4.1.1 Processo de evolução natural



- Indivíduos com habilidade de reprodução.
- Existe uma população desses indivíduos.
- Existe alguma variedade de indivíduos.
- Há diferenças na capacidade de sobrevivência dos indivíduos em seu ambiente.

Figura 1 - Seleção natural
Fonte: http://www.deti.ufc.br/~pimentel/disciplinas/ica.htm

4.1.2 Componentes de um algoritmo genético



- Um problema para ser resolvido pelo algoritmo.
- Um método para codificar soluções do problema através de cromossomos.
- Uma função de avaliação que mede quão bem cada solução é capaz de resolver o problema.
- Um método para criar a população inicial de cromossomos.
- Um conjunto de parâmetros para o algoritmo genético.

Figura 2 - Componente do Algoritmo Genético
Fonte: http://www.deti.ufc.br/~pimentel/disciplinas/ica.htm

4.1.3 Principais conceitos

Podemos dizer que os Algoritmos Genéticos fazem parte de modelos computacionais vistos como otimizadores de funções. Para a conclusão dessa pesquisa, nos baseamos em parâmetros que integram nosso problema, onde a solução é dada se encontramos um ponto satisfatório entre eles.

Seus principais conceitos são:

- **Cromossomo:** No coração do algoritmo genético está o cromossomo. O cromossomo representa uma solução potencial e é dividido em múltiplos genes.
- **Gene:** Genes em AG representam aspecaso de testeos distintos da solução como um todo. Tal qual os genes humanos

representam aspecaso de testeos distintos de indivíduos, como sexo ou cor dos olhos.

- **Fenótipo:** cromossomo codificado.
- **População:** conjunto de pontos (indivíduos) no espaço de busca.
- **Geração**: Iteração completa do AG que gera uma nova população.
- ***Fitness*:** Função de classificação, na qual é atribuída uma nota aos indivíduos.

4.1.4 Operações básicas de um algoritmo genético

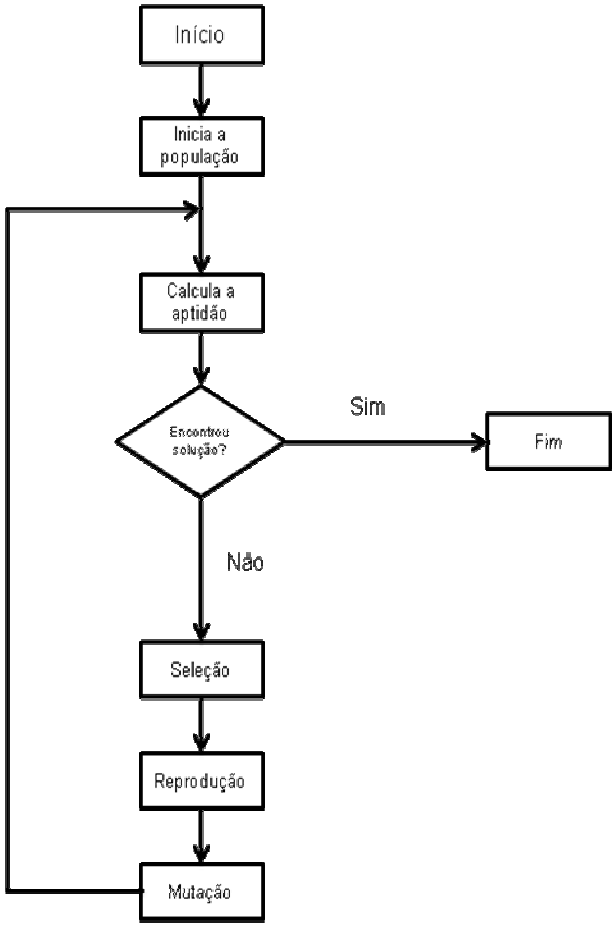


Figura 3 - Operações básicas do Algoritmo Genético
Fonte: http://www.gta.ufrj.br/~marcio/figura2.gif

4.1.5 Codificação do problema para AG

A codificação do problema foi elaborada em cinco principais etapas: planejar o cromossomo, programar a função de "*fitness*", configurar os nossos objetos, criar uma população inicial de potenciais soluções e evoluir esta população.

Neste contexto a aplicação utiliza uma API em Java, a JGAP, a qual oferece as estruturas necessárias para essa modelagem, fornecendo os mecanismos básicos para a utilização dos AGs além de permitir a customização dos mesmos.

A JGAP foi escolhida por sua flexibilidade no tratamento de AG além de todo suporte *on-line* que ela dispõe em sua página na internet. Nela encontram-se tutoriais de como utilizá-la, além de exemplos já implantados de AG.

5 METODOLOGIA

5.1 PLANEJAMENTO DO CROMOSSOMO

O cromossomo, como explicado anteriormente, é o coração do algoritmo, e é composto por um conjunto de genes. Em nosso caso, temos uma imagem que apresenta um conjunto de *spots*, onde cada um deles é um cromossomo, que será tratado como um grupo de imagens. Em vermelho, na imagem a seguir, destacamos um gene:

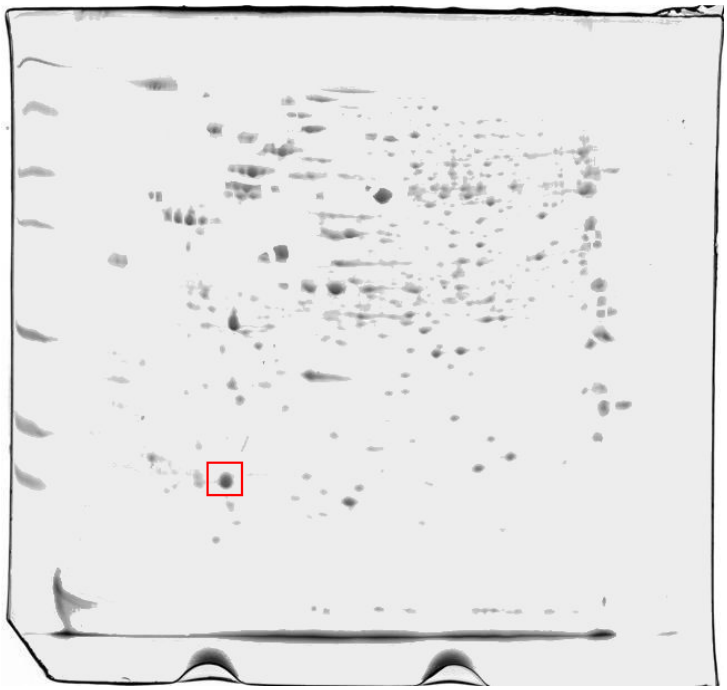


Figura 4 - Gel 2D
Fonte: Setor de Bioquímica - UFPR

Os genes que compõe o cromossomo são as coordenadas do ponto inicial do quadrado e o tamanho desse conforme a imagem a seguir:

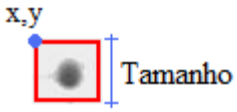


Figura 5 – Cromossomo com seus *genes* em azul
Fonte: Trabalho de Conclusão de Curso

É aplicado um pré-filtro antes do processamento da imagem para encontrar os spots, este filtro também utiliza AG e seu cromossomo é composto pelos genes com os valores de RGB do rastro e do *spot,* conforme imagem a seguir:

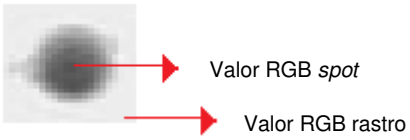


Figura 6 – Corte com sinalização dos valores do cromossomo
Fonte: Trabalho de Conclusão de Curso

Para tratar a diferença de gradiente da imagem do gel, a imagem é recortada em pedaços proporcionais ao seu tamanho, a partir de testes realizados pela equipe em mais de cinqüenta géis o corte selecionado foi de 50 linhas X 50 colunas, pois esse apresentou os melhores resultados.

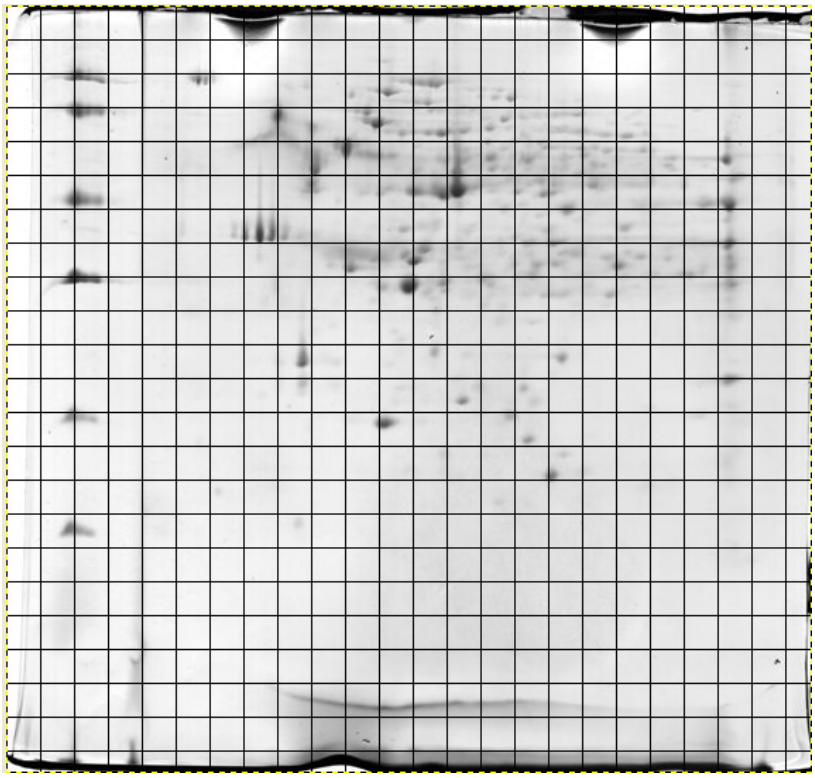


Figura 7 – Representação não exata dos cortes da imagem
Fonte: Setor de Bioquímica - UFPR

5.2 FUNÇÃO "*FITNESS*"

Cada um dos cromossomos apresentados anteriormente é classificado para o processo de cruzamento, o diagrama a seguir representa como essa função atribui uma nota a qual será usada para seleção dos indivíduos.

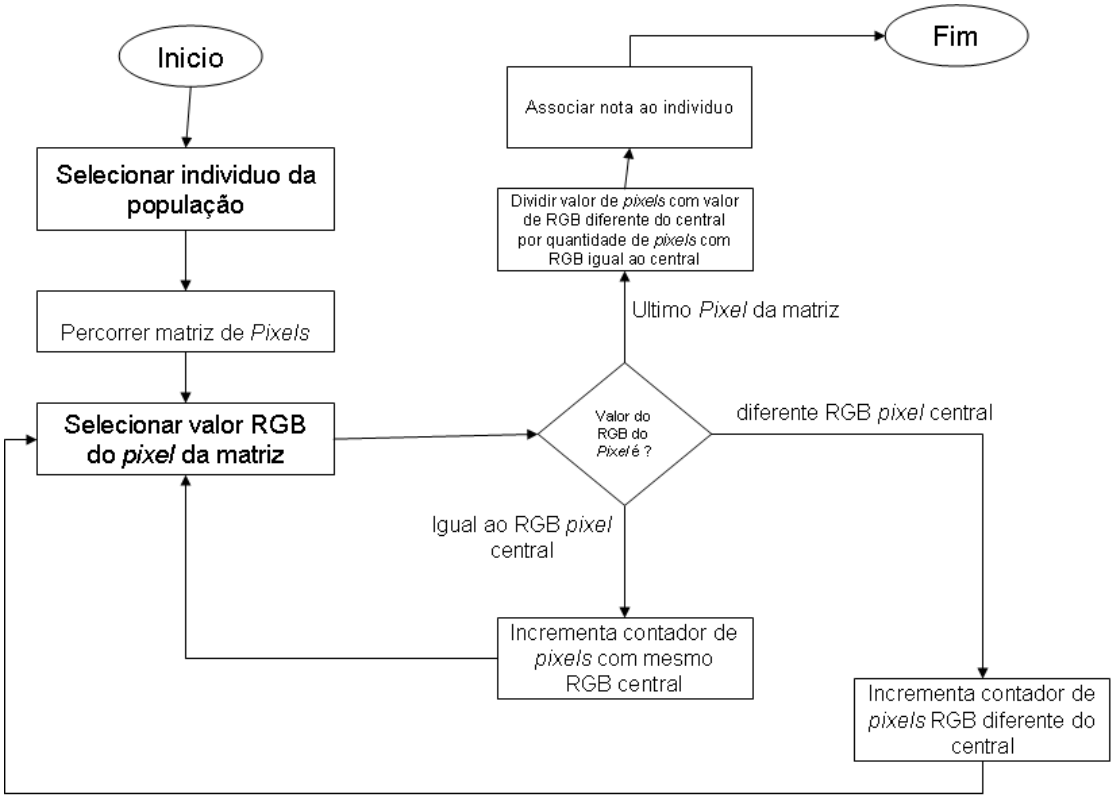
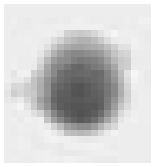


Figura 8 – Diagrama que representa a função responsável pela nota.
Fonte: Trabalho de Conclusão de Curso

Observe como seria a nota de dois indivíduos opostos: a imagem A apresenta um *spot* e a imagem B apenas rastro:



A

B

Primeiramente vamos tratar a imagem A, nela o pixel central tem valor de RGB de 85 e existem 11 pixels com esta mesma cor, logo a sua nota de acordo com a *fitness* seria de 4. Observe os cálculos detalhadamente abaixo:

x centro = 11; Y centro = 12
Valor RGB centro = 85
Quantidade de pixels com RGB 85 = 11
Tamanho da Imagem = 528
Cálculo = (528/11) * 0,1
Nota: 4

Observe que o valor do centro da imagem é sempre aproximado para um valor inteiro, assim garantindo que será comparado o RGB de apenas um pixel. Para a imagem B, quando utilizamos à mesma função classificatória essa recebe uma nota menor. Observe os cálculos abaixo:

x centro = 11; Y centro = 12
Valor RGB centro = 128
Quantidade de pixels com RGB 128 = 242
Tamanho da Imagem = 506
Calculo = (506/242) * 0,1
Nota: 0.2

Observando os dois cálculos é possível notar que os *spots* se mantêm com nota mais alta do que os rastros, assim o AG dá mais condições à imagem A de cruzar do que à imagem B.

No fluxograma a seguir representamos a função de *fitness* dos filtros:

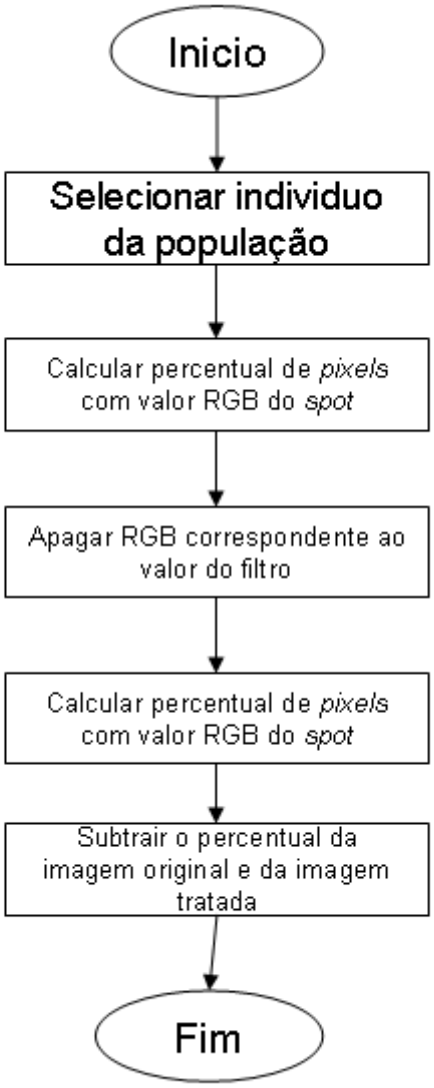


Figura 9 - Função *Fitness*
Fonte: Trabalho de Conclusão de Curso

Na classificação apresentada a seguir observamos que a figura 10 é a imagem original, figura 11 a imagem com um tratamento onde o spot é corrompido e a figura 12 os *spots* não apresentam alterações. Salientando que neste algoritmo não são encontrados os spots, apenas é apagada uma parte do que poderá ser rastro ou apenas fundo da imagem.

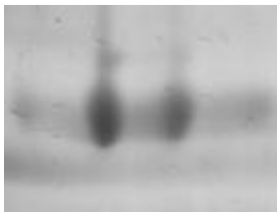


Figura 10 - Imagem original
Fonte: Setor de Bioquímica – UFPR

Quando a "*fitness"* trata a imagem com dois filtros de 145 e 200, observamos o resultado a seguir:

Figura 11 - Imagem com RGBs maiores que 145 apagados
Fonte: Setor de Bioquímica – UFPR

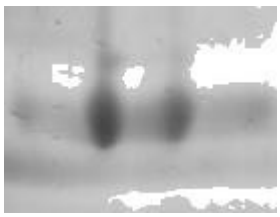


Figura 12 - Imagem com RGBs maiores que 200 apagados
Fonte: Setor de Bioquímica - UFPR

Na figura 11, observamos a imagem tratada com o filtro de 145, que recebeu nota de -30, já na figura 12 a imagem tratada com o filtro de 200 que recebe nota de -34, sendo assim o filtro de 145 teria mais chances de evoluir. Este algoritmo funciona apenas como um facilitador para o segundo AG onde a imagem aqui tratada será utilizada para buscar os *spots*. Para montar a imagem tratada sempre será copiada a área considerada *spot*, sendo assim, é garantida que as propriedades da imagem utilizadas para os estudos biológicos sejam mantidas, ou seja, não altera a propriedade do *spot* na imagem.

5.2.1 Configurando a população

Para cada um dos AGs aplicados é configurada uma população inicial, a qual é feita com valores aleatórios controlados pelo JGAP, no AG de filtros a população inicial é de 50 indivíduos, que passa pelo processo de cruzamento 50 vezes e apenas o melhor indivíduo é selecionado para tratar a imagem. Para o AG que encontra os *spots* a população inicial é de 1000 indivíduos que passa pelo processo de evolução 100 vezes e os 500 melhores indivíduos são considerados os *spots*.

6 PLANEJAMENTO DO PROJETO

Segundo John L. Beckley, "A maioria das pessoas não planeja fracassar, fracassa por não planejar". O planejamento do projeto propicia segurança frente às variações que o produto possa sofrer em relação às suas especificações iniciais. Para gerenciar o projeto SOFIA foi criado um plano de projeto, o qual está estruturado com: escopo do projeto, cronograma, riscos e seus tratamentos, divisão de tarefas dentro cronograma, recursos e mecanismos de *tracking*. (O plano geral de projeto encontra-se em sua totalidade no apêndice 1).

Com a elaboração clara do escopo pode-se planejar claramente o projeto, o que foi essencial, pois o trabalho teve um foco claro com usuários, orientadores e alunos comprometidos.

O PGP auxiliou durante todo o projeto, pois com o cronograma estabelecido foi possível sincronizar as atividades que poderiam ser realizadas simultaneamente, além de alocar os recursos da melhor maneira possível. Quanto aos ajustes necessários no cronograma, o mais relevante foi à alteração da data de entrega, pois a data prevista no PGP era 23/11, a qual foi alterada para o dia 30/11. Com isso a equipe revisou toda a documentação, o que certamente contribuiu para qualidade do projeto. Outro fator interessante no decorrer do projeto foi administrar as atividades que estavam no caminho crítico, estabelecido no PERT/CPM e no Gráfico de *Gantt*, assim foi possível evitar atrasos e "realocar" mais recursos para uma tarefa quando esta demonstrava algum problema.

A clara divisão de tarefas durante o projeto evitou que um membro da equipe ficasse sobrecarregado, o que certamente contribuiu para evitar possíveis desentendimentos da equipe. Com as tarefas alinhadas ao cronograma ficou mais simples gerenciar e acompanhar as atividades e verificar como e onde cada recurso estava alocado. Os mecanismos de *tracking* auxiliaram toda a equipe a ficar informada do *status* do projeto e das atividades.

7 CONTROLE DE QUALIDADE

O plano de SQA estabelece todas as atividades da função gerencial que determinam a política da qualidade, os objetivos e as responsabilidades, o planejamento e o controle da qualidade. Determina: os padrões de código, critérios para verificação e validação dos artefatos. (O plano de SQA encontra-se em sua totalidade no apêndice 2).

É interessante estabelecer claramente os padrões de qualidade no projeto e os meios para controlá-los, pois assim fica claro que os critérios foram atingidos. Por exemplo, a aprovação de interface trouxe uma opinião interessante do usuário, desse modo foi possível validar que uma interface intuitiva realmente foi à melhor opção.

Quanto à qualidade de código, na reunião de revisão, foi possível eliminar os métodos e classes desnecessárias que haviam sido criados apenas para testes. Aperfeiçoamos alguns métodos e colocamos nomes mais significativos nas variáveis, reestruturando a fim de atender os padrões que facilitarão manutenções no sistema.

O plano de testes foi primordial para encontrar erros que os usuários poderiam cometer, e assim corrigi-los de forma apropriada antes da entrega. Um dos requisitos mais importantes do nosso projeto é a portabilidade, assim os casos de teste foram executados em varias plataformas, o que nos permitiu garantir a operação em multiplataforma. Os resultados dos testes se encontram anexos ao plano de SQA.

8 MODELAGEM DO SISTEMA

A modelagem do sistema foi feita em UML, permitiu criar um programa devidamente estruturado e com classes bem organizadas. Os diagramas utilizados para o design do sistema foram:

- Diagrama de classes (apêndice 3)
- Diagrama de casos de uso (apêndice 4)
- Diagrama de seqüência (apêndice 5)
- Diagramas de colaboração (apêndice 6)
- Diagramas de atividades (apêndice 7)
- Diagrama de pacotes (apêndice 8)
- Diagrama de componentes (apêndice 9)

Uma análise preliminar foi estabelecida no PGP onde estava prevista a criação o diagrama de telas, porém devido às caracaso de testeerísticas do sistema, foi substituído pelo diagrama de pacotes listado acima.

Os diagramas foram elaborados em duas ferramentas. O Jude foi a principal, pois ele oferece recursos de exportação de código e sua interface é de fácil manuseio. Porém como o Jude não oferece suporte ao diagrama de pacotes foi utilizado o *Rational Software Modeler* da IBM.

Os diagramas foram criados a partir dos *Use cases* (apêndice 4). Os quais foram elaborados pela gerente do projeto também responsável pela análise de requisitos. Com a análise de requisitos completa os diagramas puderam ser elaborados. Para criação do diagrama de classes foi necessário pontuar quais as principais classes do programa e como elas interagiam entre si, no entanto o diagrama foi alterado conforme a fase de desenvolvimento do sistema.

Os diagramas de atividade e de seqüência auxiliaram a manter claro qual o fluxo que deveria ser mantido pelo sistema. Além de demonstrar o funcionamento da integração entre os dois Algoritmos Genéticos utilizados no sistema.

9 INTERFACE

Ao desenvolver interfaces de forma ergonômica, podemos nos deparar com algumas dificuldades devido ao comportamento variável dos usuários de *software*, visto o ambiente tecnológico em constante evolução que nos encontramos. Conforme o contexto em que eles se encontram as entradas e saídas podem representar coisas diferentes. A usabilidade representa a união entre 4 pontos: interface, tarefa, usuário e ambiente.

Dentro do trabalho apresentado buscamos tornar o sistema agradável, com facilidade de ser operado, intuitivo e eficiente. Para alcançar os objetivos procuramos: a identificação do que o sistema precisava em termos de entradas e saídas de dados; realizamos a conferência para ver se as exigências dos usuários foram cumpridas; corrigimos a solução apresentada inicialmente; estabelecemos critérios de ergonomia e produzimos a solução de interface do projeto.

O autor Bem Shneiderman traz oito “regras de ouro” para projetar e avaliar interfaces no seu livro *Designing the user interface* (SHNEIDERMAN & PLAISANT, 2004):

- Perseguir a consistência;
- Fornecer atalhos;
- Fornecer *feedback* informativo;
- Marcar o final dos diálogos;
- Fornecer prevenção e manipulação simples de erros;
- Permitir o cancelamento das ações;

- Fornecer controle e iniciativa ao usuário;
- Reduzir a carga de memória de trabalho.

Baseado nos estudos realizados utilizamos os seguintes critérios ergonômicos nas interfaces do sistema:

CONDUÇÃO: visa fornecer o aprendizado para utilização por pessoas que não estão habituadas com o sistema. Onde analisamos por: (FAUST, 2007)

Convite	Títulos claros para: telas, janelas e caixas de diálogo.
Agrupamento e distinção de itens	Lista de dados coesa e itens na tela separados por relações lógicas.
Legibilidade	Textos em contrastes efetivos com o fundo.
Feedback imediato	Relata as informações recebidas e indicação que haverá demora no processamento solicitado.

Tabela 1 - Critérios ergonômicos
Fonte: Trabalho de Conclusão de Curso

AÇÕES MÍNIMAS: minimiza o conjunto de ações para alcançar seu objetivo. Ex: não solicitar dados que o programa pode gerar. Não solicitar o mesmo dado várias vezes. (FAUST, 2007)

PROTEÇÃO CONTRA ERROS: visa detecaso de testear e prevenir os possíveis erros dos usuários. Ex: detecaso de testear erros nas entradas de dados. (FAUST, 2007)

QUALIDADE NAS MENSAGENS DE ERRO: apresentar mensagens de forma que o usuário possa compreender tendo diferentes níveis de conhecimento. Ex: indicar o que ele fez de errado e o que deve fazer. Ser breve. (FAUST, 2007)

10 RESULTADOS

Com a finalização do projeto de pesquisa foram alcançados os objetivos propostos seguindo os requisitos apresentados pelo departamento de bioquímica da UFPR. A seguir a síntese dos resultados obtidos após os testes do *software* SOFIA (apêndice 2) e seu uso no *ImageMaster™ 2D Platinum*:

REQUISITOS DO SOFIA	Sim	Não
Apresenta interface simples e intuitiva?	x	
É multiplataforma?	x	
Elimina os ruídos da imagem do gel 2D?	x	
Mantém as propriedades do spot na imagem do gel 2D?	x	
Disponibiliza a imagem nos formatos necessários?	x	
Apresenta gráficos em 3D dos *spots*?		X
É possível adicionar *spots* perdidos com o processamento?	x	
É possível eliminar rastros não apagados com o processamento?	x	

REQUISITOS DE USO NO *IMAGEMASTER™ 2D PLATINUM*	Sim	Não
Reconhece os formatos das imagens salvas pelo SOFIA?	x	
Importa imagem do SOFIA corretamente?	x	
Detecta os *spots* nas imagens pré-processadas pelo SOFIA?	x*	
Realiza os cálculos dos *spots* nas imagens pré-processadas pelo SOFIA?	x	
É possível gerar gráficos a partir das imagens pré-processadas pelo SOFIA?	x	

* Conforme os testes realizados no *software ImageMaster™ 2D Platinum* com a imagem do gel 2D pré-processada pelo SOFIA, obtivemos uma melhora na detecção do spots, onde em comparação com a imagem não processada ele encontra mais *spots* "verdadeiros". Por exemplo: em uma imagem onde o *software ImageMaster™ 2D Platinum* detecta 1000 *spots*, sendo eles apenas 300 verdadeiros, com esta mesma imagem pré-processada no SOFIA ele detecta 250 verdadeiros. Concluímos que ainda são necessários alguns ajustes, que podem ser encontrados no capítulo 12.

11 CONCLUSÃO

Com a criação do *software* SOFIA a equipe teve a oportunidade de conhecer a demanda de *software*s para o departamento de Bioquímica da UFPR, entender o processo de eletroforese e sua importância para o setor.

Utilizando técnicas de Inteligência Artificial (através de Algoritmos Genéticos) e técnicas de Visão Computacional, o processamento das imagens de géis de eletroforese 2D pelo *software* desenvolvido tornou-se mais eficiente, pois os *spots* foram encontrados rapidamente e sem muitos erros.

Por ser desenvolvido em Java, utilizando a plataforma Eclipse, o projeto pode ser exportado para diversas plataformas e executado normalmente em diversas versões: como *XP* e *Seven* do *Windows*, *Ubuntu* do *Linux* e *OpenSolaris* da *Sun OS*.

De acordo com o plano de SQA pode-se garantir a qualidade dos processos de produção do *software*. Garantindo os seguintes critérios: qualidade de interface, qualidade de operação em plataformas, qualidade de código e qualidade de operação.

De acordo com o PGP as estimativas de risco, cronograma, recursos, organização, mecanismos de *tracking* e controle foram seguidos, tornando a execução do projeto mais ágil. Este plano de projeto foi elaborado para o desenvolvimento do sistema SOFIA , com fundamentação acadêmica e intuito de obter a graduação, e apresenta as estimativas de projeto, os riscos, o cronograma, os recursos do projeto, a organização de pessoas, mecanismos de *tracking* (rastreamento) e controle.

Através da análise UML o projeto pode ser demonstrado e estudado para garantir que todas as funcionalidades foram corretamente implantadas.

O *software* SOFIA é capaz de minimizar os ruídos das imagens de géis de eletroforese 2D sem perder as propriedades da imagem, deixando somente os *spots*, e permite salvar em vários formatos. Dessa maneira os pesquisadores do setor de Bioquímica da UFPR poderão usufruir da interface intuitiva do *software* e importar a imagem pré-processada para o *software* que desejarem.

12 PROJETOS FUTUROS

Durante a etapa final do projeto foram identificados vários pontos de melhoria, os quais serão implementados futuramente, pois o nosso cronograma já estava estabelecido de acordo com o escopo inicial. Os pontos de implementação futura são:

- Implementação da funcionalidade de *zoom* simultâneo nas imagens.
- Implementação de um gráfico 3D referente ao volume dos *spots*.
- Implementação de uma rede para a classificação mais precisa dos *spots*.
- Migração para o ambiente virtual.

13 BIBLIOGRAFIA

BLACKSTOCK, W.P.; WEIR, M.P. **Proteomics:** quantitative and physical mapping of cellular proteins. Reviews: TIBTECH. , v.17, p.121-127, 1999.

FAUST, Adriana Holtz Betiol Richard. **Ergonomia e Usabilidade:** Conhecimentos, métodos e aplicações. São Paulo: NOVATEC, 2007.

JOCELYN, H.; Ilag, L. **Dividing and conquering proteomics**. Proteomics , p.96-98, 2002.

RAFAEL C. Gonzalez; RICHARD E. Woods. **Processamento de imagens digitais.** São Paulo: Edgard Blücher LTDA, 2000.

SHNEIDERMAN Ben; PLAISANT Catherine. **Designing The User Interface**. Boston: Pearson, 2005.

STUART Russell; PETER Nowing. **Inteligência Artificial.** Rio de Janeiro: Campus, 2004

APÊNDICE 1 – PLANO DE GERENCIAMENTO DE PROJETOS (PGP)

SUMÁRIO

1 INTRODUÇÃO

Este plano de projeto foi elaborado para o desenvolvimento do sistema SOFIA, apresenta as estimativas, os riscos, o cronograma e os recursos do projeto além da organização de pessoas, mecanismos de *tracking* (rastreamento) e controle.

2 ESCOPO E PROPÓSITO DO DOCUMENTO

Este plano de projeto visa apresentar o planejamento para o desenvolvimento da aplicação SOFIA, e está estruturado da seguinte forma:

Objetivos do projeto: onde são definidos os objetivos do projeto, suas principais funções, questões de desempenho e restrições técnicas e administrativas;

Estimativas de projeto: onde são definidas as estimativas do projeto e as técnicas e dados históricos utilizadas para definir essas estimativas;

Riscos do projeto: Subdividido em duas partes:

- **Análise de Riscos:** onde os riscos são identificados, as estimativas são feitas e avaliadas.
- **Administração dos Riscos:** onde são definidas opções para evitar os riscos e procedimentos de monitoração dos mesmos;

Cronograma: onde é feita a divisão do trabalho no projeto (*Work Breakdown Strucaso de testeure*), as redes de tarefas, gráficos de *Gantt* e tabela de recursos necessários (pessoal);

Recursos do projeto: onde são definidos os recursos que utilizaremos no projeto, tanto pessoal e de *hardware*/*software*, como recursos especiais que podem ser requeridos para uma boa condução do projeto;

Organização do pessoal: onde são definidos a estrutura da equipe e os relatórios administrativos a serem emitidos durante o projeto;

Mecanismos de *tracking* (rastreamento) e controle: onde são definidos os meios de controle, bem como esse processo irá ocorrer durante as várias etapas do projeto;

3 OBJETIVOS DO PROJETO

Desenvolver e implantar um *software* capaz de efetuar o pré-processamento das imagens digitais de géis 2D resultantes das pesquisas proteômicas, utilizando técnicas de inteligência artificial, tendo como meta a diminuição dos ruídos presentes nas imagens resultantes da corrida do gel, facilitando a detecção dos *spots*.

3.1 FUNÇÕES PRINCIPAIS

O sistema tem como principais as seguintes funções:

1. Carregamento de imagens em vários formatos;
2. Processamento da imagem;
3. Exibição da imagem pré-processada;
4. Possibilidade de o usuário comparar imagem carregada e depois de processada, permitindo a cópia de parte da imagem caso o usuário note a perda de algum *spot* no pré-processamento;
5. Salvar as imagens processadas em diversos formatos.
6. Exibição de gráficos.

3.2 QUESTÕES DE DESEMPENHO

O cronograma deverá ser seguido rigorosamente para garantir o cumprimento das tarefas no prazo. Esse controle deve ser feito pela gerente do projeto.

Haverá uma divisão do projeto em fases para melhor desempenho da equipe e também reuniões semanais para dúvidas a serem sanadas, verificação do andamento do trabalho realizado pelos integrantes e avaliação dos objetivos para verificar se estão sendo alcançados.

O projeto será mostrado ao professor orientador quinzenalmente para avaliação. A qualidade é muito importante dentro do nosso contexto de desenvolvimento, para isso utilizaremos técnicas de revisão, e testes durante o processo.

3.3 RESTRIÇÕES TÉCNICAS E ADMINISTRATIVAS

Professor orientador: disponibilidade de tempo para avaliação e orientação das fases concluídas do projeto;

3.4 METODOLOGIA DE DESENVOLVIMENTO

Para o desenvolvimento do sistema utilizaremos o modelo *Extreme Programing* (XP).

Este sistema valoriza as pessoas envolvidas no processo, possibilitando um desenvolvimento mais rápido devido à programação em par, onde os programadores trabalham em duplas com auxilio mútuo, de tal forma que o código apresenta mais qualidade, facilitando a manutenção.

Juntamente com o modelo XP utilizaremos a metodologia de ciclo de vida em Cascata, pois nesta o desenvolvimento é feito em fases bem definidas, onde uma fase só prossegue quando a anterior termina. Porém são previstas alterações pontuais em alguns artefatos produzidos em fases anteriores para adaptação as mudanças de visão do projeto como, por exemplo, o modelo conceitual do diagrama de classes elaborado na fase de análise poderá ser alterado na fase de desenvolvimento.

4 ESTIMATIVA DE TEMPO

Usamos como estimativa de tempo o ano letivo de aula, começando em Março/2009 e se encerrando em Novembro/2009, que será o tempo necessário para desenvolvimento e entrega do *software*.

5 RISCOS DO PROJETO

5.1 ANÁLISE DE RISCOS

Os riscos do projeto são os resultados de estudos feitos a partir da probabilidade de uma determinada atividade falhar de acordo com o que foi previamente planejado. Somente o fato de uma atividade existir, existe possibilidades de ocorrerem eventos vantajosos para cima dela ou eventos que ameaçam o sucesso. O gerenciamento dos riscos é essencial, para que no decorrer do projeto haja a mitigação das conseqüências dos riscos.

5.2 IDENTIFICAÇÃO DOS RISCOS

De acordo com as delimitações do projeto, os riscos possíveis foram listados, levando em consideração fatores técnicos, relacionamento humano, relações externas e contratuais.

Riscos de acordo com o ciclo de vida do projeto e áreas do PMBOK.

Início

1. Definição errada do escopo do projeto.
2. Orientador ausente.

Planejamento

1. Definição errada das atividades a serem desenvolvidas (atraso no cronograma).
2. Definição errada do tempo necessário para conclusão das atividades (atraso no cronograma).
3. Alocação falha dos recursos necessários para concluir determinada atividade.
4. Problemas de comunicação entre os integrantes ou com o cliente.

Execução

1. Falta de luz em alguma ocasião (na programação).
2. Desentendimentos entre os participantes do time ou com o cliente.
3. Ausência de algum integrante do time (por motivos de doença, morte, abandono do projeto).
4. Utilização dos laboratórios da Escola Técnica ser interrompida por fatores não controláveis.
5. Dificuldade pelos integrantes do time no manuseio das ferramentas necessárias.

Controle

1. Mudanças no escopo.

Encerramento

1. Usuário ou professor orientador não concordar com o que foi feito.

Riscos externos que afetam o projeto

1. Ações governamentais que prejudiquem o andamento do projeto.

Para estimar os riscos, deve-se quantificá-los e qualificá-los. Uma matriz de probabilidade dos riscos pode identificar se o risco tem baixa, média ou alta probabilidade de ocorrer. Essas probabilidades se relacionam com os dias de atraso de cada atividade do WBS e com os riscos previamente identificados.

5.3 ANÁLISE QUALITATIVA DOS RISCOS

A análise qualitativa foi o processo realizado para avaliar o impacto do risco reconhecido e sua probabilidade numérica de ocorrência.

Para avaliação da probabilidade de ocorrência foi usada a tabela extraída do livro "*Managing Risk*", SEI, com algumas adaptações.

Pontuação	Probabilidade percebida	Probabilidade (1 a 5)	Probabilidade (%)
Muito baixo	As chances são insignificantes	1	Menos que 20%
	É muito improvável		
	Não há praticamente chance nenhuma		
Baixo	Pouca chance	2	Menos que 40%
	Provavelmente não acontecerá		
	Improvável		
Moderado	Pouco provável	3	Menos que 60%
	Existem dúvidas		
	Mais ou menos		
Alto	Achamos que sim	4	Menos que 80%
	Provavelmente não acontecerá		
	Presumível		
Muito alto	As chances são consideráveis	5	Menos que 100%
	Muito provável		
	É praticamente certo		

Figura 13 - Tabela usada na definição da probabilidade de ocorrência
Fonte: "*Managing Risk*", SEI

Para medir o impacto foi utilizada a seguinte classificação:

Número	Impacto
1	Muito baixo
2	Baixo
3	Moderado
4	Alto
5	Muito alto

Figura 14 - Tabela de impacto
Fonte: *"Managing Risk"*, SEI

Probabilidade x Impacto:

Para saber a probabilidade de um risco, usou-se a escala de probabilidade, que vai de 1 a 5. Uma probabilidade 1 significa que há total certeza que o risco não irá ocorrer e uma probabilidade 5 significa que há total certeza que o risco irá ocorrer.

Probabilidade X Impacto					Classificação (*)		
Ciclo de Vida	Risco nº	Descrição	Prob. (1 a 5)	Imp. (1 a 5)	Alto	Médio	Baixo
Início	1	Definição errada do escopo do projeto	4	4	X		
	2	Orientador ausente	2	3		X	
Planejamento	3	Definição errada das atividades a serem desenvolvidas	5	5	X		
	4	Definição errada do tempo necessário para conclusão das atividades	4	5	X		
	5	Alocação falha dos recursos necessários para concluir determinada atividade	4	3	X		
	6	Problemas de comunicação entre os integrantes ou com o cliente	2	2		X	
Execução	7	Falta de luz em alguma ocasião	2	3		X	
	8	Desentendimentos entre os participantes do time ou com o cliente	5	3	X		
	9	Ausência de algum integrante do time	2	2		X	
	10	Utilização dos laboratórios da ET ser interrompida por fatores ñ controláveis	1	2			X
	11	Dificuldade pelos integrantes do time no manuseio das ferramentas necessárias.	3	2		X	
Controle	12	Mudanças no escopo	5	4	X		
Encerramento	13	Cliente ou professor orientador não concordar com o que foi feito.	4	5	X		
Externo	14	Ações governamentais que prejudiquem o andamento do projeto	1	4		X	

(*) Classificação: é o produto das colunas Probabilidade e Impacto
De 1 a 3 = Baixo -> requer apenas controle
De 4 a 6 = Médio -> há necessidade de monitoração ativa e redução do risco, quando possível
Maior que 7 = Alto -> é necessário alguma ação para reduzir o risco

Figura 15 - Probabilidade X Impacto
Fonte: *"Managing Risk"*, SEI

Tendo por base a tabela de classificação dos riscos (probabilidade x impacto) foi elaborado o plano de gerenciamento desses riscos.

5.4 ANÁLISE QUANTITATIVA REPRESENTADA EM NÚMEROS

De acordo com os riscos identificados e qualificados foi realizada a quantificação, através de uma tabela avaliativa das opções disponíveis, caso seja efetivo. A tabela tem por objetivo facilitar a visualização das melhores ações a serem tomadas para contornar o problema, evidenciando o objetivo principal do projeto. É uma árvore de decisão transposta em forma de tabela.

	Risco	Possibilidade	Descrição	Ganho (1 a 5)
1	Definição errada do escopo do projeto	1	adequar o projeto	5
		2	seguir do jeito que está	1
2	Orientador ausente	1	procurar alguém que possa substituí-lo	5
3	Definição errada das atividades a serem desenvolvidas	1	refazer a definição das atividades	3
4	Definição errada do tempo necessário para conclusão das atividades	1	refazer o cronograma	2
5	Alocação falha dos recursos necessários para concluir determinada atividade	1	realocar integrantes ans atividades	4
6	Problemas de comunicação entre os integrantes ou com o cliente	1	implantar novos metodos de comunicação	5
7	Falta de luz em alguma ocasião	1	trocar de local	5
8	Desentendimentos entre os participantes do time ou com o cliente	1	restabelecer a ordem	5
9	Ausência de algum integrante do time	1	refazer o cronograma	4
10	Utilização dos laboratórios da ET ser interrompida por fatores ñ controláveis	1	trocar de local	5
11	Dificuldade pelos integrantes do time no manuseio das ferramentas necessárias.	1	empenho dos integrantes no estudo	5
		2	estabelecer contato com prof. Da area	5
12	Mudanças no escopo	1	adequar o projeto	2
13	Cliente ou professor orientador não concordar com o que foi feito.	1	não há o que fazer	1
14	Ações governamentais que prejudiquem o andamento do projeto	1	adequar o projeto	3

Figura 16 - Análise quantitativa
Fonte: *"Managing Risk"*, SEI

Avaliação dos riscos encontrados

Os riscos encontrados são em grande maioria de classificação média e alta, tendo apenas dois riscos baixos. Porém, as respostas aos riscos foram determinadas na maioria dos casos, ou seja, caso esses riscos sejam efetivos há alternativa para dar continuidade ao projeto, o que o torna viável se analisado pelos riscos que o envolvem.

Administração dos riscos

Para administrar os riscos, em casos que é possível evitá-los, foi proposto medidas para tal. Na maioria dos casos que isso não é possível, foi proposto medidas para administrar os riscos de forma segura, sem comprometer o andamento do projeto.

Procedimentos de monitoração dos riscos

Os riscos foram listados de acordo com as fases do ciclo de vida de um projeto (iniciação, planejamento, execução, controle e encerramento) justamente para ficar claro o momento em que eles devem ser monitorados e de que forma. Essa colocação facilita a visibilidade das fases e os fatores críticos, as fases de maior risco e que se deve monitorar assiduamente. Nas reuniões semanais com o time deverá ser feita a avaliação dos riscos de acordo com o andamento do projeto e caso novos fatores de vulnerabilidade forem identificados, serão documentados e incluídos no Plano de Gerenciamento de Riscos.

5.5 GERENCIAMENTO DE RISCOS.

Definição errada do escopo do projeto.

- Definição do risco: definir de forma incorreta o que terá que ser feito para atingir o objetivo final do projeto.
- Conseqüência de sua ocorrência: a definição errada do escopo compromete todo o projeto, pois tudo foi estruturado tendo como base o escopo.
- Opções para evitar o risco: executar reuniões com os integrantes do time e com o cliente, antes do projeto iniciar, para discutir o problema e o que de fato será necessário para concluí-lo.
- Resposta ao risco (ações corretivas): se o risco for efetivado tentar adaptar o escopo correto no tempo restante e reutilizar o que já foi feito (se isso for possível).

Professor Orientador ausente.

- Definição do risco: falta do professor orientador por motivos de doença, abandono do projeto, etc.

- Conseqüência de sua ocorrência: a falta de avaliação do trabalho feito e auxílio em questões técnicas podem propiciar atrasos no cronograma e nas atividades.
- Opções para evitar o risco: neste caso não há como se evitar, e sim, apenas aceitar, pois não é algo que se possa controlar, por ser um projeto acadêmico.
- Resposta ao risco (ações corretivas): uma alternativa seria procurar o co-orientador ou um professor que possa realizar o papel e repassar as responsabilidades a ele.

Definição errada das atividades a serem desenvolvidas (atraso no cronograma).

- Definição do risco: definição defeituosa das atividades necessárias para concluir os trabalhos, devido à falta de experiência do time.
- Conseqüência de sua ocorrência: se isso realmente acontecer, atrasará todo cronograma, além de causar retrabalho.
- Opções para evitar o risco: realizar reuniões com profissionais da área mais experientes (professores, por exemplo) para evitar o risco.
- Resposta ao risco (ações corretivas): refazer a definição de atividades de forma correta e o mais rápido possível reiniciar os trabalhos.

Definição errada do tempo necessário para conclusão das atividades (atraso no cronograma).

- Definição do risco: estimar incorretamente o tempo que levará para conclusão das atividades, principalmente pela falta de experiência do time.
- Conseqüência de sua ocorrência: atrasará o cronograma, cliente insatisfeito, time começa a se desgastar, pode gerar conflitos internos.
- Opções para evitar o risco: realizar reuniões com profissionais da área (mais experientes) para que isso não ocorra.
- Resposta ao risco (ações corretivas): refazer o cronograma.

Alocação falha dos recursos necessários para concluir determinada atividade.

- Definição do risco: definir de forma errada os participantes do time nas atividades a serem realizadas. Colocar em uma tarefa simples mais profissionais do que o necessário.
- Conseqüência de sua ocorrência: se isso acontecer, causará atraso em algumas atividades, comprometendo a qualidade do trabalho desenvolvido e o cronograma do projeto.
- Opções para evitar o risco: conversar com profissionais da área (mais experientes), pesquisar em projetos de mesmo porte para tentar evitar o risco.
- Resposta ao risco (ações corretivas): realocar os integrantes nas atividades mais demoradas e trabalhosas e adequar o cronograma para que não haja atraso nas entregas ou pelo menos, minimizar esse atraso.

Problemas de comunicação entre os integrantes ou com o usuário.

- Definição do risco: falha na comunicação entre os integrantes do time ou com o usuário.
- Conseqüência de sua ocorrência: a falta de comunicação pode gerar desentendimentos, atraso no cronograma e redundância nos trabalhos.
- Opções para evitar o risco: através dos mecanismos de rastreamento, com reuniões com o usuário e com o time, informação sendo repassada de forma direta, clara e para todos os envolvidos.
- Resposta ao risco (ações corretivas): caso não esteja acontecendo uma boa comunicação, implantar mecanismos mais eficazes de comunicação.

***Blackout* em alguma ocasião (na programação).**

- Definição do risco: na fase de execução, faltar luz nos laboratórios.
- Conseqüência de sua ocorrência: se isso ocorrer, atrasará os trabalhos de implementação e o projeto todo fica vulnerável.
- Opções para evitar o risco: por ser um trabalho acadêmico e não envolver custos monetários, não há o que fazer apenas aceitar o risco, trabalhar com

suas conseqüências e responder satisfatoriamente (plano de contingência), pois qualquer solução nesse sentido sairia caro (geradores, por exemplo).

- Resposta ao risco (ações corretivas): uma solução caso isso ocorra, seria transferir os trabalhos para outro local, no caso, a casa de cada uma das integrantes, tendo em vista que todos da equipe possuem computador pessoal com condições para o desenvolvimento.

Desentendimentos entre os participantes do time ou com o usuário.

- Definição do risco: problemas pessoais entre os integrantes do time ou com o cliente (cliente não colaborar).
- Conseqüência de sua ocorrência: pode ser catastrófico para o projeto, pois quem o faz acontecer são as pessoas e desentendimentos entre elas podem atrasar cronograma, atividades, prejudicar a qualidade ou na pior das hipóteses algum integrante abandonar o projeto, desistir. Já com o usuário pode causar reprovação na matéria, tendo em vista que a parte de modelagem será feita para obtenção de nota em uma das matérias do curso, a parte de implementação fica por conta das integrantes que optaram por continuar com o produto.
- Opções para evitar o risco: nesse caso, não há o que fazer para evitar, apenas aceitar o risco.
- Resposta ao risco (ações corretivas): caberá a gerente de projeto restabelecer a ordem e o respeito entre os integrantes e gerenciar os prejuízos, da mesma forma se os problemas forem com o cliente.

Ausência de algum integrante do time (por motivos de doença, morte, abandono do projeto).

- Definição do risco: por motivos diversos, como doença, morte, afastamento do curso, alguma integrante poderá se ausentar do projeto.
- Conseqüência de sua ocorrência: implicará em sobrecarga de trabalho para os outros participantes do time e terá que ser adequado o Plano Geral de Projeto de acordo com o período de ausência do indivíduo.

- Opções para evitar o risco: não tem como evitar, apenas aceitar, pois não é algo controlável.
- Resposta ao risco (ações corretivas): refazer o planejamento e adequar com a situação atual, reorganizar os trabalhos.

Utilização dos laboratórios da Escola Técnica ser interrompida por fatores não controláveis.

- Definição do risco: bloqueio de utilização dos laboratórios da Escola Técnica.
- Conseqüência de sua ocorrência: prejudicará as fases de análise, implementação e testes.
- Opções para evitar o risco: não há o que fazer, apenas aceita-lo.
- Resposta ao risco (ações corretivas): transferir os trabalhos para outro local, no caso, as integrantes possuem computadores pessoais com as ferramentas necessárias para dar continuidade ao projeto.

Dificuldade pelos integrantes do time no manuseio das ferramentas necessárias.

- Definição do risco: pela pouca experiência, poderá ocorrer dificuldade no manuseio das ferramentas, principalmente nas fases de implementação e testes.
- Conseqüência de sua ocorrência: atraso no cronograma.
- Opções para evitar o risco: contatar profissionais da área que possuam domínio das ferramentas e obter o máximo de conhecimento (treinamento).
- Resposta ao risco (ações corretivas): participantes terão que se empenhar para superar essa dificuldade, através de livros, pesquisas na internet, contatar colegas de profissão com o devido conhecimento.

Mudanças no escopo.

- Definição do risco: alterações no escopo do projeto.

- Conseqüência de sua ocorrência: como as tarefas foram planejadas em torno do escopo, alterações nesse sentido podem prejudicar todos os planos do projeto.
- Opções para evitar o risco: definir bem o escopo, fazendo reuniões com o cliente, tentando compreender ao máximo o que será necessário fazer para atender o objetivo do projeto. Resposta ao risco (ações corretivas): adequar o Plano Geral de Projeto de acordo com as alterações de escopo, tentando reduzir o prejuízo.
- Resposta ao risco (ações corretivas): adequar o projeto à nova situação.

Usuário ou professor orientador não concordar com o que foi feito.

- Definição do risco: usuário não aceitar o que foi feito.
- Conseqüência de sua ocorrência: por ser um projeto de conclusão de curso, a conseqüência seria a reprovação do time.
- Opções para evitar o risco: reuniões periódicas com o cliente e professor orientador, a fim de sincronizar as idéias.
- Resposta ao risco (ações corretivas): não há o que fazer.

Ações governamentais que prejudiquem o andamento do projeto.

- Definição do risco: definições políticas que de alguma forma impeçam ou prejudiquem o andamento do projeto.
- Conseqüência de sua ocorrência: dependendo da ação governamental, influenciará as áreas do projeto ou, na pior das situações o abortará.
- Opções para evitar o risco: não há o que fazer, apenas aceitar.
- Resposta ao risco (ações corretivas): adequar o plano de acordo com a nova situação, tentando minimizar os prejuízos.

6 CRONOGRAMA

O cronograma do projeto apresenta uma lista de tarefas com as etapas bem definidas. Devido a nossa metodologia de trabalho de cascata, para demonstrarmos o cronograma, apresentamos o WBS (*Work BreakDown Structure*), o diagrama de *Gantt* e a rede de tarefas com o caminho crítico do projeto.

6.1 *WORK BREAKDOWN STRUCTURE* (WBS)

Para demonstrar a disposição das fases e as suas tarefas, foi elaborado o WBS, que contém as atividades que serão desenvolvidas ao longo do projeto, que será dividido em 4 fases, conforme mostra a figura abaixo:

Figura 17 - WBS
Fonte: Trabalho de Conclusão de curso

6.2 REDE DE TAREFAS

O PERT/CPM foi gerado a partir da ferramenta *Microsoft Office Project de teste 2007* e indica o caminho crítico do planejamento das atividades em função do tempo de duração de cada uma, recursos utilizados e definição de inter-relacionamentos, possibilitando uma visão geral do caminho que poderia atrasar todo o cronograma. O que se pode observar como críticos são as fases de modelagem, implementação e um pouco do módulo de testes e correção. A especificação do sistema também foi apontada no caminho, pois começará juntamente com o início do projeto (passando também pelos pontos críticos), ou seja, à medida que conclui os trabalhos especifica-se a parte correspondente da documentação.

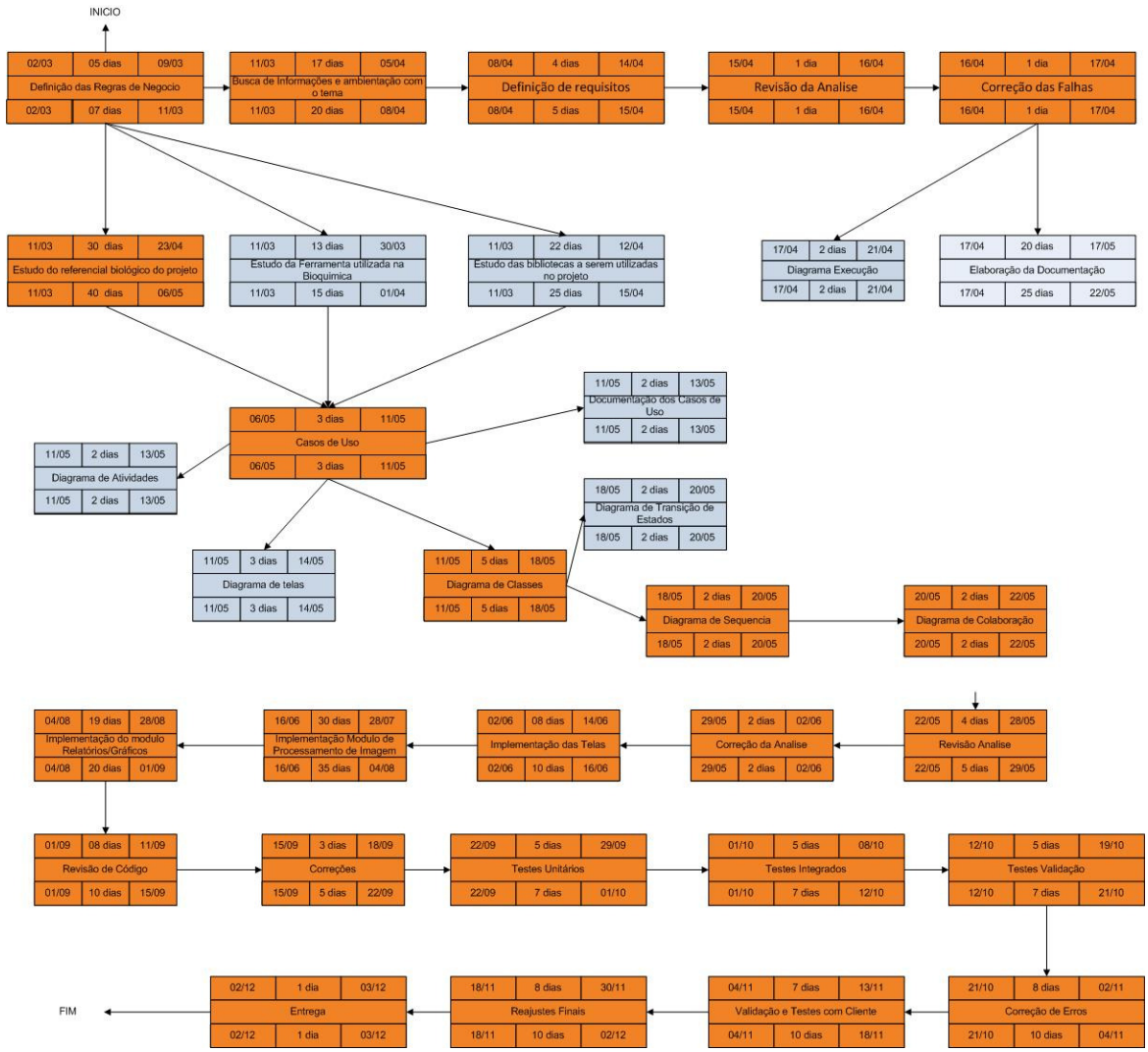


Figura 18 - PERT/CPM
Fonte: Trabalho de Conclusão de curso

6.3 GRÁFICO DE *GANTT*

O Gráfico de *Gantt* (encontra-se no anexo 1) mostra de forma sucinta a alocação dos recursos de acordo com as atividades e o tempo em que elas ocorrem.

6.4 TABELAS DE RECURSOS

A tabela de recursos ficou dividida em 3 partes, que são:

- Declaração de Escopo
- Tabela de Atividades
- Tabela de Recurso x Cronograma

6.4.1 Declaração de Escopo

A declaração de escopo aborda todas as atividades que terão de ser cumpridas para a obtenção do produto final. Apresenta uma descrição do que será realizado em cada atividade citada.

FASE	NOME DA TAREFA	DESCRIÇÃO
Negócios	Definição das Regras de Negocio	Verificar e definir o que é necessário ser feito no projeto
	Busca de Informações e ambientação com o tema	Pesquisar sobre o tema, ambiente atual e outras informações necessárias ao projeto
	Definição de requisitos	Definir com o cliente e com o professor orientador tudo que precisa ser feito
	Revisão da Analise	Rever tudo que foi estudado e pesquisado
	Correção das Falhas	Corrigir inadequações encontradas
	Estudo da Ferramenta utilizada na Bioquimica	Estudar a ferramenta disponível hoje, e meios de melhora-la
	Estudo do referencial biológico do projeto	Pesquisar e estudar artigos sobre o tema
	Estudo das bibliotecas a serem utilizadas no projeto	Estudo da linguagem JAVA e biblioteca JGAP
Análise	Casos de uso	Construção dos casos de uso do sistema
	Documentação De casos de uso	Documentação escrita
	Diagrama de telas	Construir diagrama
	Diagrama de Classes	
	Diagrama de seqüência	
	Diagrama de Transição de Estados	
	Diagrama de atividades	
	Diagrama De Execução	
	Diagrama de colaboração	
	Revisão analise	Rever todos os diagramas e verificar se contemplam todo o projeto
	Correção de falhas	Corrigir falhas encontradas nos diagramas
Implementação	Elaboração Da documentação	Elaborar documentação com todos os diagramas
	Implementação das telas	Elaboração das telas, botões e imagens
	Implementação do modulo de Processamento da Imagem	Codificação, uso das bibliotecas e da linguagem JAVA para processar e limpar a imagem
	Implementação do modulo Relatórios/Gráficos	Codificação, uso das bibliotecas e da linguagem JAVA para gerar graficos
	Revisão	Revisão da Implementação
	Correções da Revisão	Corrigir possíveis erros encontrados
	Testes Unitários	Testes unitários de acordo com os casos de uso
	Testes Integrados	Testes do sistema todo
	Testes Validação	Testes de validação
	Correções de Erros	Corrigir erros encontrados nos testes
	Validação e Testes com cliente	Reunião de validação do sistema com o cliente
Implantação	Reajustes Finais	Reajustes solicitados
	Entrega	Finalização do projeto

Figura 19 - Declaração de escopo
Fonte: Trabalho de Conclusão de curso

6.4.2 Tabela de Atividades

Na tabela de atividades foi exposto, além das atividades, o tempo de duração previsto para cada uma delas com suas respecaso de testeivas predecessores e o inter-relacionamento entre elas, a fim de melhor gerenciar as dependências de cada uma, pois refletirá diretamente no tempo do projeto.

NOME DA TAREFA	DURAÇÃO	PREDECESSORAS
Definição das Regras de Negocio	7 dias	
Busca de Informações e ambientação com o tema	20 dias	1
Definição de requisitos	5 dias	1;2
Revisão da Analise	1 dia	1;2;3
Correção das Falhas	1 dia	4
Estudo da Ferramenta utilizada na Bioquimica	15 dias	1
Estudo do referencial biológico do projeto	40 dias	1
Estudo das bibliotecas a serem utilizadas no projeto	25 dias	1
Casos de uso	3 dias	5;6;7;8
Documentação De casos de uso	2 dias	9
Diagrama de telas	3 dias	9
Diagrama de Classes	5 dias	9
Diagrama de seqüência	2 dias	12
Diagrama de Transição de Estados	2 dias	12
Diagrama de atividades	2 dias	9
Diagrama De Execução	2 dias	5
Diagrama de colaboração	2 dias	13
Revisão analise	5 dias	9;10;11;12;13;14;15;16;17
Correção de falhas	2 dias	18
Elaboração Da documentação	25 dias	5
Implementação das telas	10 dias	20
Implementação do modulo de Processamento da Imagem	35 dias	20;21
Implementação do modulo Relatórios/Gráficos	20 dias	22;21
Revisão	10 dias	23
Correções da Revisão	5 dias	24
Testes Unitários	7 dias	25
Testes Integrados	7 dias	25;26
Testes Validação	7 dias	25;27
Correções de Erros	10 dias	26;28;27
Validação e Testes com cliente	10 dias	29
Reajustes Finais	10 dias	29;30
Entrega	1 dia	31

Figura 20 - Atividades
Fonte: Trabalho de Conclusão de curso

6.4.3 Tabela de Recurso x Cronograma

Os recursos foram definidos de acordo com a complexidade das atividades, experiência dos integrantes da equipe, ferramentas de auxílio e preferências dos mesmos. Em alguns casos, foi definido mais de um integrante para execução. O início e fim das atividades são de suma importância para o gerenciamento de tempo e elaboração da rede de tarefas.

NOME DA TAREFA	DURAÇÃO	DATA INICIAL	DATA FINAL	PREDECESSORAS	NOME DO RECURSO
Definição das Regras de Negocio	7 dias	2/3/2009 09:00	11/3/2009 09:00		Equipe
Busca de Informações e ambientação com o tema	20 dias	11/3/2009 09:00	8/4/2009 09:00	1	Equipe
Definição de requisitos	5 dias	8/4/2009 09:00	15/4/2009 09:00	1;2	Equipe
Revisão da Analise	1 dia	15/4/2009 09:00	16/4/2009 09:00	1;2;3	Angélica;Letícia
Correção das Falhas	1 dia	16/4/2009 09:00	17/4/2009 09:00	4	Aline
Estudo da Ferramenta utilizada na Bioquimica	15 dias	11/3/2009 09:00	1/4/2009 09:00	1	Equipe
Estudo do referencial biológico do projeto	40 dias	11/3/2009 09:00	6/5/2009 09:00	1	Equipe
Estudo das bibliotecas a serem utilizadas no projeto	25 dias	11/3/2009 09:00	15/4/2009 09:00	1	Equipe
Casos de uso	3 dias	6/5/2009 09:00	11/5/2009 09:00	5;6;7;8	Equipe
Documentação De casos de uso	2 dias	11/5/2009 09:00	13/5/2009 09:00	9	Angélica
Diagrama de telas	3 dias	11/5/2009 09:00	14/5/2009 09:00	9	Aline
Diagrama de Classes	5 dias	11/5/2009 09:00	18/5/2009 09:00	9	Aline
Diagrama de seqüência	2 dias	18/5/2009 09:00	20/5/2009 09:00	12	Vanely
Diagrama de Transição de Estados	2 dias	18/5/2009 09:00	20/5/2009 09:00	12	Juliana
Diagrama de atividades	2 dias	11/5/2009 09:00	13/5/2009 09:00	9	Letícia
Diagrama De Execução	2 dias	17/4/2009 09:00	21/4/2009 09:00	5	Juliana
Diagrama de colaboração	2 dias	20/5/2009 09:00	22/5/2009 09:00	13	Aline
Revisão analise	5 dias	22/5/2009 09:00	29/5/2009 09:00	9;10;11;12;13;14;15;16;17	Equipe
Correção de falhas	2 dias	29/5/2009 09:00	2/6/2009 09:00	18	Equipe
Elaboração Da documentação	25 dias	17/4/2009 09:00	22/5/2009 09:00	5	Angélica;Juliana;Letícia;Vanely
Implementação das telas	10 dias	22/5/2009 09:00	5/6/2009 09:00	20	Equipe
Implementação do modulo de Processamento da Imagem	35 dias	5/6/2009 09:00	24/7/2009 09:00	20;21	Equipe
Implementação do modulo Relatórios/Gráficos	20 dias	24/7/2009 09:00	21/8/2009 09:00	22;21	Equipe
Revisão	10 dias	21/8/2009 09:00	4/9/2009 09:00	23	Equipe
Correções da Revisão	5 dias	4/9/2009 09:00	11/9/2009 09:00	24	Angélica
Testes Unitários	7 dias	11/9/2009 09:00	22/9/2009 09:00	25	Juliana
Testes Integrados	7 dias	22/9/2009 09:00	1/10/2009 09:00	25;26	Aline
Testes Validação	7 dias	1/10/2009 09:00	12/10/2009 09:00	25;27	Juliana
Correções de Erros	10 dias	12/10/2009 09:00	26/10/2009 09:00	26;28;27	Equipe
Validação e Testes com cliente	10 dias	26/10/2009 09:00	9/11/2009 09:00	29	Equipe
Reajustes Finais	10 dias	9/11/2009 09:00	23/11/2009 09:00	29;30	Equipe
Entrega	1 dia	23/11/2009 09:00	24/11/2009 09:00	31	Equipe

Figura 21 - Recursos X Cronograma
Fonte: Trabalho de Conclusão de curso

7 RECURSOS DO PROJETO

Custos não foram calculados devido ao fato de o projeto ser com fundamentação acadêmica, e ser utilizados recursos próprios de *hardware* (além, dos computadores da própria universidade) e *software*s gratuitos ou com licença proprietária.

7.1 PESSOAL

Aline Ariane Schultz

Angélica Inajá Juliani

Juliana Helena Tibães

Letícia Fernandes de Jesus

Vanely de Souza

7.2 HARDWARE

Pentium IV

1.0 GHz

HD 80Gb

1Gb de Memória RAM

7.3 SOFTWARE

Ferramenta de Programação – Eclipse – IDE para desenvolvimento de aplicações em Java.

Ferramenta de Testes Unitários – Junit

Ferramenta Case para elaboração dos Diagramas – *Jude UML Community*

Ferramenta de Gerenciamento de Projetos – *MSProjecaso de teste*

Ferramenta de Gerenciamento de Projetos – *WBS Chart Pro*

Ferramenta de Controle de Versões – *Subversion*

Ferramenta de Documentação de Código-Fonte – JavaDoc

Ferramenta de Ambiente de Testes (homologação) – *IBM Rational Test*

8 ORGANIZAÇÃO DO PESSOAL

8.1 ESTRUTURA DE EQUIPE

Com o total de 5 membros da equipe, definimos as tarefas de cada integrante:

- Angélica Inajá Juliani: Gerente do Projeto. Também responsável por criar e documentar os casos de uso, elaboração da documentação, fazer as revisões, além da implementação do sistema.

- Aline Ariane Schultz: responsável pela documentação dos casos de uso, diagramas e testes integrados, além da implementação do sistema.

- Juliana Helena Tibães: responsável por diagramas, elaboração da documentação, testes e implementação do sistema.

- Letícia Fernandes de Jesus: responsável pela elaboração da documentação, diagramas, revisões e implementação do sistema.

- Vanely de Souza: responsável pela elaboração da documentação, diagramas, estudos aprofundados na área de biologia, diagramas e implementação do sistema.

8.2 RELATÓRIOS ADMINISTRATIVOS

Definimos que serão feitos relatórios a cada reunião com a equipe e com o orientador. As atas serão elaboradas por um dos membros da equipe que ficará responsável também por enviá-las por e-mail a todos os demais. As atas deverão seguir o modelo demonstrado abaixo.

	Sofia – Modelo de ata de Reunião de acompanhamento
Elaborado por:	
Período:	
Atividades:	
Andamento:	
Pontos de ação:	
Obs.	

9 MECANISMOS DE *TRACKING* (RASTREAMENTO) E CONTROLE

Tendo como base os relatórios administrativos gerados, serão realizadas reuniões com a equipe de execução do projeto, com intuito de avaliar:

- Andamento das atividades
- Cumprimento do cronograma
- Dificuldades técnicas e pessoais enfrentadas
- Controle de riscos
- Verificação do caminho crítico
- Necessidade de alterações e impacaso de testeo causado
- Exposição de novos riscos encontrados

A cada mês haverá reunião com o professor orientador. Será lhe apresentado o andamento do projeto, de resultados, esclarecimento de dúvidas e será definido o rumo das próximas atividades.

ANEXO 1

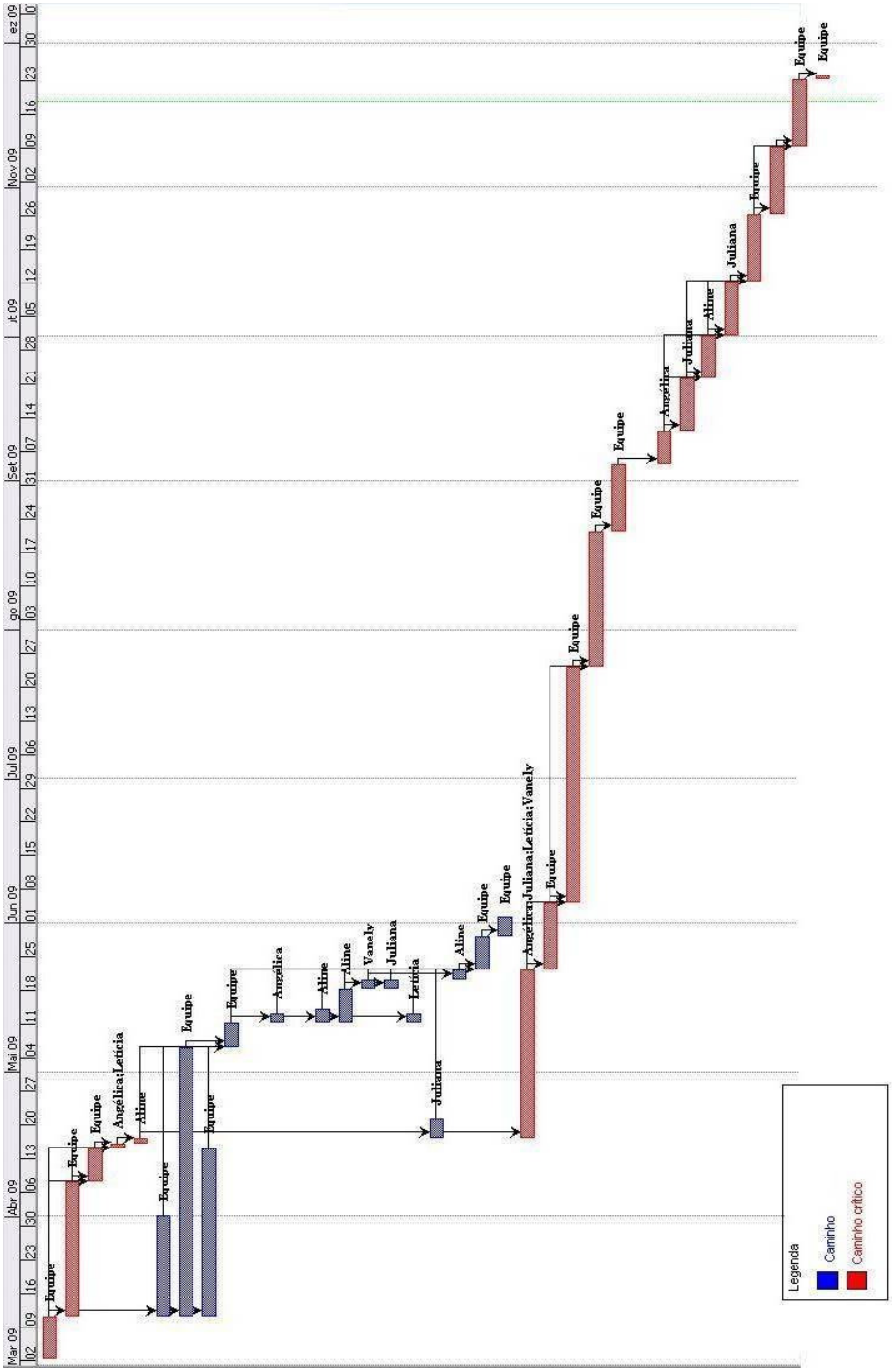


Figura 22 - Gráfico de *Gantt*
Fonte: Trabalho de Conclusão de curso

10 BIBLIOGRAFIA

BELLI, M. J. **Estimativas**. Apresentação em PowerPoint.

BELLI, M. J. **Estudo de Riscos**. Apresentação em PowerPoint.

FÁBRICA de *Software* III. Disponível em: <http://www.cin.ufpe.br/~fabrica3/homePage/planoProcesso.htm#elaborarWBS>. Acesso em 27 abr. 2009.

GERENCIAMENTO de Projetos. Disponível em: <http://www.projetoe.org.br/vteams/teles/tele_02/leitura_02_3.html>. Acesso em 27 abr. 2009.

HALL, Elaine M. **Managing Risk:** Methods for Software Systems Development.1998

MARTINS, J. C. C. **Gestão de projetos de desenvolvimento de *software* (PMI - UML)**. Rio de Janeiro: Brasport, 2002.

TECNOLOGIA de Projetos. Disponível em: <http://www.tecnologiadeprojetos.com.br>. Acesso em 27 abr. 2009.

VARGAS, R. V. **Gerenciamento de Projetos:** estabelecendo diferenciais competitivos. Rio de Janeiro: Brasport, 2005.

APÊNDICE 2 – PLANO DE SQA

SUMÁRIO

1 INTRODUÇÃO

A mudança tecnológica que ocorreu na década de 70 em relação aos computadores teve um efeito dramático na produção de *software*, pois, num pequeno período de tempo a capacidade das máquinas aumentou muito e com isso passou-se a exigir mais potencial de *software*. Os projetos ficaram mais complexos e sem muito controle. As companhias de desenvolvimento apresentavam dificuldades de entregar os *software*s dentro do tempo, orçamento e da qualidade. Os projetos entregues nessa época eram desastrosos, pois o orçamento era excedido, atrasos constantes e qualidade pobre, inaceitável. (ANDRÉ KOSCIANSKI & MICHEL DOS SANTOS SOARES,2007).

O que marcou os anos 80 foi a "crise do *software*". Mas juntamente com essa crise surgiu também a solução: a engenharia de *software.* Um dos primeiros registros sobre o assunto foi em 1968, na Alemanha em uma conferência, curiosamente organizada por uma entidade que aparentemente não tinha nenhuma ligação com a área: o comitê de ciência da NATO (*North Atlantic Treaty Organization*). A partir disto vários órgãos passaram a se organizar a fim de ajudar os desenvolvedores a melhorar a qualidade de processo de produção de *software*. (ANDRÉ KOSCIANSKI & MICHEL DOS SANTOS SOARES, *SOFTWARE* ENGENEERING INSTITUTE, 2007).

Nessa época foi criado o SQA, evoluído a partir do IV&V, tornou-se uma ferramenta eficaz nas companhias de desenvolvimento de *software*, ajudando a identificar problemas da qualidade mais cedo no processo de desenvolvimento, o que, entre outros benefícios, torna o custo com reparos, muito menor.

A SEI (*Software Engeneering Institute*) define o Plano de SQA (Garantia da qualidade de *software*) como um planejado e sistemático padrão de todas as ações necessárias para prover adequada relação de que um *software* produto de trabalho, *"work producaso de teste"*, está em conformidade com os requisitos técnicos estabelecidos. Ao afirmar isso, se parte do princípio que a equipe de SQA deve estar completamente interada de todo o processo para cumprir seus objetivos, dando suporte a equipe de desenvolvimento em todas as fases da criação do produto, garantindo com o maior custo beneficio um bom grau de qualidade no processo e no produto final.

Nesse mesmo contexto foram surgindo os modelos de maturidade, dentre os quais podemos citar o CMM, que futuramente serviu de base para o CMMI.

O *Capability Maturity Model Integration* (CMMI) é um guia para melhoria dos processos nas organizações. Sendo assim podemos dizer que um plano de SQA pode utilizar o CMMI como base. O CMMI fornece orientações para criação de praticas e objetivos. Ele também fornece parâmetros para que a empresa possa avaliar e divulgar o nível de capacidade que se encontra o processo.

O documento a seguir define o plano de SQA, com a finalidade de apresentar um produto final com qualidade de *software*.

2 DEFINIÇÃO DO PROJETO

2.1 OBJETIVOS

O projeto SOFIA desenvolve um *software* com IA, utilizando a técnica de Algoritmo Genético. Este *software* ira tratar uma imagem de gel de eletroforese 2D e eliminar dessas os rastros deixando apenas os *spots*.

2.2 FUNÇÕES PRINCIPAIS

O sistema tem as funções de carregar imagem, processar imagem, fazer ajustes manuais e mostrar gráficos que demonstrem que a imagem não perdeu propriedades.

2.3 QUESTÕES DE DESEMPENHO

Por ser um *software* de tratamento de imagem a questão de desempenho se torna critica, pois normalmente *software*s que fazem este tratamento são lentos. O *software* desenvolvido trabalhará em escala com as imagens para aperfeiçoar os processos.

2.4 RESTRIÇÕES TÉCNICAS E ADMINISTRATIVAS

Quanto à utilização de recursos de *hardware* o sistema deverá ser “leve” para possibilitar seu uso em qualquer maquina.

3 OBJETIVOS DO PLANO DE SQA

Este plano visa garantir qualidade dos processos de produção do *software*, adotando o conceito de CRUSBY (1992), ”Qualidade é a conformidade com os requisitos”. Neste projeto o cliente é o departamento de bioquímica da UFPR sendo assim é necessária uma atenção especial quanto à portabilidade do sistema, já que o governo brasileiro e a UFPR incentivam o uso de *software*s livres. De acordo com o conceito citado acima se estabelecem os seguintes critérios de qualidade do projeto:

3.1 QUALIDADE DE INTERFACE

De acordo com requisitos estabelecidos, as interfaces devem ser simples, objetivas e conterem um link para o manual do usuário. Para garantir que essas exigências sejam seguidas segundo os padrões do cliente, haverá uma reunião de aprovação de interfaces.

3.2 QUALIDADE DE OPERAÇÃO EM PLATAFORMAS

É estabelecido que o *software* deva executar tanto em plataforma livre e/ou fechada. Podendo ser utilizada em diversas configurações de *hardware*.

3.3 QUALIDADE DE CÓDIGO

O sistema deve seguir o padrão pré-estabelecido no capítulo de Padrões.

3.4 QUALIDADE DE OPERAÇÃO

O processamento de uma imagem de gel não deverá demorar mais que 2 minutos.

4 DOCUMENTAÇÃO

Será contado como documentação todo o produto da análise aos relatórios dos testes, além do Plano Geral de Projetos (PGP) e este documento. Diagramas da analise:

- Diagrama de classes (apêndice 3)
- Diagrama de casos de uso (apêndice 4)
- Diagrama de seqüência (apêndice 5)
- Diagramas de colaboração (apêndice 6)
- Diagramas de atividades (apêndice 7)
- Diagrama de pacotes (apêndice 8)
- Diagrama de componentes (apêndice 9)

5 PADRÕES, PRÁTICAS, CONVENÇÕES E MÉTRICAS

5.1 PADRÕES

Programar é uma tarefa intelectual, por isso encontrar uma solução de um problema é feito individualmente, mas em um ambiente de desenvolvimento, onde há o trabalho em equipe é preciso a cooperação da mesma para o sucesso do projeto.

Porém como estabelecer esse canal de comunicação, onde todos dentro da equipe e o próprio programador possam ter controle dos códigos? Visando melhorar a nossa qualidade de código, e reaproveitamento do mesmo, se estabelece aqui uma serie de padrões de código para nosso projeto.

Linguagem Utilizada	Java

Comentário inicial

Toda classe deve possuir um comentário inicial, contendo uma descrição, versão, autor, conforme modelo abaixo:

```
/**
 *
 * <p><b>
 * Descrição da classe e sua função/Responsabilidade
 * </b></p>
 * @author Equipe de desenvolvimento Sofia
 * @version
 */
```

Declaração de métodos

Os métodos deverão ser precedidos de um comentário padrão, seguindo o modelo a seguir:

```
/**
 *
 * <p><b>
 * Descrição do metodo.
 * </b></p>
 *
 * @param Lista de parametros com curta descrição
 * @throws
 */
```

Identação

Deve ser utilizada a tecla “tab” para identação.

Comentários

Os comentários deverão ser moderados dentro dos blocos de código. É importantíssimo se manter os padrões estabelecidos no início da classe e dos métodos. Para os atributos das classes deverá ser colocado um bloco padrão antes da declaração da mesma da seguinte forma:

```
    /**
     * atributo contendo o separador de tipos de arquivo
     */
            public static final String DIVISOR_ARQUIVOS = ", ";
```

Este padrão fica estabelecido para facilitar a geração do JavaDoc. Para os demais comentários dentro dos blocos de códigos são estabelecidos os padrões a seguir:

Comentário de linha única

Um breve comentário pode aparecer em uma única linha identado ao nível do código que se segue, e deve ser precedido por uma linha em branco:

```
if (condição) {

    / * Manuseie o estado. */
    ... ...
 }
```

Comentário Rasteiro

Um comentário muito breve pode aparecer na mesma linha do código que descreve, mas devem ser transferidos a uma distância suficiente para separá-lo do código:

```
if (a == 2) {
    return TRUE;            /* special case */
} else {
    return isPrime(a);      /* works only for odd a */
}
```

Comentário de Final de linha

As duas barras (//) devem ser usadas para comentário fora uma linha de código completa ou parcialmente (inativar as linhas de código). Não deve ser usado em várias linhas consecutivas para textos de observações, porém, pode ser usado para comentar as seções de código. Exemplos de todos os três estilos seguir:

```
if (foo > 1) {

    // Do a double-flip.
    ...
}
else {
    return false;          // Explain why here.
}
//if (bar > 1) {
//
//    // Do a triple-flip.
//    ...
//}
//else {
// return false;
//}
```

Declaração de número por linha

É recomendada uma declaração de uma variável por linha:

```
int level; // indentation level
int size;  // size of table
```

Não coloque diferentes tipos na mesma linha. Exemplo:

int foo, fooarray[]; / / ERRADO!

Declaração de inicialização

As variáveis são inicializadas nos construtores das classes ou nos métodos que as utilizarão.

Convenções de nomenclaturas

Convenções de nomenclatura tornam os programas mais compreensíveis, facilitando a leitura. Elas também podem dar informações sobre a função do identificador, por exemplo, se é um pacote, uma constante, ou uma classe, além de que podem ser úteis para a compreensão do código.

Tipo Identificador	Regras para Nomeação	Exemplos
Classes	Os nomes devem ser substantivos. No caso de mais de uma palavra, a primeira letra da segunda palavra deve ser maiúscula. Tente manter os nomes de classe simples e descritivos. Procure evitar siglas e abreviaturas.	ClassRaster; class ImageSprite;

Métodos	Métodos devem ser verbos, com todas as letras minúsculas. No caso de mais de uma palavra, a primeira letra da segunda palavra deve ser maiúscula.	run();
		runFast();
		getBackground();

Variáveis	Exceto para as variáveis, todas as instâncias, classe e constantes da classe estão em minúsculas e, no caso de mais de uma palavra, a primeira letra da segunda palavra deve ser maiúscula. Os nomes de variáveis não devem começar com sublinhado _ ou sinal de dólar $, embora ambos sejam permitidos. Os nomes de variáveis devem ser curtos, mas significativos. A escolha de um nome de variável deve ser um mnemônico, isto é, destinada a indicar para o observador casual, a intenção do seu uso. Nomes de variáveis com apenas um caracaso de testeer devem ser evitados, exceto para as variáveis temporárias ou "descartáveis". Os nomes comuns para as variáveis temporárias são **i, j, k, m** e **n** para inteiros**, c, d** e para caracaso de testeeres.	int i; char c; float myWidth;

Constantes	Os nomes de variáveis declaradas constantes de classe devem estar totalmente em letras maiúsculas com as palavras separadas por sublinhados ("_").	static final int MIN_WIDTH = 4;
		static final int MAX_WIDTH = 999;
		static final int GET_THE_CPU = 1;

6 REVISÕES

A revisão da documentação será realizada em reuniões semanais da equipe. Nessas reuniões cada artefato produzido é apresentado aos demais membros que avaliarão os documentos.

6.1 REVISÃO DAS REGRAS DE NEGÓCIO

As regras de negócio serão validadas com o usuário em uma reunião de aprovação.

6.2 REVISÃO DE CÓDIGO

A revisão de código será realizada antes da entrega do mesmo, em duas reuniões semelhantes às de aprovação de documentação, porém, nesta será montada uma apresentação aos outros membros do time que observarão tanto padrões do código quanto possíveis otimizações. Estas ficarão documentadas em um relatório diferenciado especifico para esta reunião. Será preenchido um *checklist* (em anexo neste documento) para cada classe do sistema.

7 TESTES

O plano de testes descreve o planejamento dos testes que serão realizados ao longo do desenvolvimento do *software*. Deverá incluir os tipos de testes necessários para garantir a conformidade do produto de *software* com os requisitos estabelecidos pelo escopo do projeto.

O processo de testar um programa ocorre com a intenção de encontrar um erro. Dentro deste contexto são elaborados casos de testes para verificar todas as funções. Os testes devem descobrir diferentes classes de erros em um montante mínimo de tempo e com um mínimo de esforço. Um benefício secundário de testes é que ela demonstra que o *software* está funcionando como indicado nas especificações. Os dados recolhidos através de testes também podem fornecer uma indicação da confiabilidade do *software* e de qualidade. Sendo assim definimos como escopo dos testes do nosso projeto:

- Identificar informações do projeto e os componentes de *software* que devem ser testados;
- Listar os requisitos e testá-los;
- Identificar os recursos necessários e prover uma estimativa dos esforços de teste;
- Listar os elementos resultantes do projeto de teste.

7.1 TESTES UNITÁRIOS

Unidade é a menor coleção de código que pode ser testada. Em nosso projeto, as menores partes a serem testadas são as classes os métodos. Testes de unidade são apenas um dos níveis de testes. O teste de unidade é geralmente visto como uma "caixa branca", ou seja, é inclinado a olhar e avaliar o código da aplicação, ao invés de avaliar a conformidade com algum conjunto de requisitos.

Os testes unitários deverão garantir que: estruturas de dados locais sejam examinadas para assegurar que os dados armazenados temporariamente mantenham sua integridade durante todas as etapas de execução do algoritmo, os

limites sejam estabelecidos para limitar ou restringir o processamento e todos os caminhos independentes através da estrutura de controle sejam cobertos. Assim como os erros de manipulação dos caminhos estejam cobertos.

7.2 TESTE DE VALIDAÇÃO

Validação do *software* é conseguida através de uma série de testes de “caixa preta” que vão demonstrar a conformidade com os requisitos. Tanto o plano de teste quanto os procedimentos de teste devem ser projetados para assegurar que todos os requisitos funcionais, comportamentais e de desempenho sejam satisfeitos. Neste projeto serão elaborados casos de teste para cada um dos fluxos descritos nos UCs. Estes casos de teste estão em anexo a este documento. Os testes serão realizados em vários sistemas operacionais para validação multiplataforma.

ANEXO 1

REVISÃO DE ARTEFATO – MODELO

Nome do Artefato:	
Data:	
Participantes:	

Assuntos discutidos

_____
_____
_____
_____

Pontos de melhoria no artefato

_____
_____
_____

Pontos de Ação

Atividade:	
Responsável :	

CHECKILIST DE CÓDIGO

Os nomes de variáveis estão no padrão adequado?	
O nome da classe está no padrão adequado?	
O padrão de identação foi seguido?	
O código está claro?	
Os comentários são significativos?	
Os nomes de variáveis são significativos?	
As linhas não ultrapassam 80 caracaso de testeeres?	
Em linhas gerais a nota para este código é?	

Pontos de melhoria no código.

_____

_____

_____

_____

_____

Pontos de Ação

Atividade:	
Responsável:	

ATA REUNIÃO DE APROVAÇÃO DE INTERFACE

(ata preenchida após a realização da reunião em 01/11/2009)

Você consegue identificar as funções do programa apenas olhando?	Sim
O tempo de resposta do programa é adequado?	Sim
As figuras seguem um padrão?	Sim
As figuras estão apropriadas?	Sim
É possível identificar alguma dificuldade de utilização?	Não
A interface atende as suas expecaso de testeativas?	Sim
Existem pontos de melhoria?	Sim

Pontos de melhoria na interface:

Segundo o professor Luciano, do departamento de Bioquímica: “Seria interessante ter um *zoom* na imagem para que o usuário possa selecionar um treco da imagem e avaliar, além de um gráfico em 3D da imagem.” (sic)

Pontos de Ação

Atividade:	Implementar o *zoom*
Responsável:	A definir.

ANEXO 2 – CASOS DE TESTE

(Resultados preenchidos após realização dos testes no período de novembro e dezembro)

Caso de Teste 1

Sistema Operacional: Windows 7

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: Carregar imagem

Procedimento para execução:

1 - Abrir o sistema.
2 - Clicar no botão abrir.
3 - Procurar uma nova imagem no computador.

Resultado esperado: o sistema deve carregar a imagem e exibi-la no *software*.

Resultado obtido: imagem abriu corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 2

Sistema Operacional: Ubuntu *Desktop* 9.09

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: Carregar imagem

Procedimento para execução:

1 - Abrir o sistema.

2 - Clicar no botão abrir.

3 - Procurar uma nova imagem no computador.

Resultado esperado: o sistema deve carregar a imagem e exibi-la no *software*.

Resultado obtido: imagem abriu corretamente

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 3

Sistema Operacional: OpenSolaris

Data da Realização: 01 de dezembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: Carregar imagem

Procedimento para execução:

1 - Abrir o sistema.

2 - Clicar no botão abrir.

3 - Procurar uma nova imagem no computador.

Resultado esperado: o sistema deve carregar a imagem e exibi-la no *software*.

Resultado obtido: imagem abriu corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 4

Sistema Operacional: Windows XP

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: equipe

Versão: Sofia.jar versão 1.0

Descrição: Carregar imagem

Procedimento para execução:

1 - Abrir o sistema.
2 - Clicar no botão abrir.
3 - Procurar uma nova imagem no computador.

Resultado esperado: o sistema deve carregar a imagem e exibi-la no *software*.

Resultado obtido: imagem abriu corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 5

Sistema Operacional: Windows 7

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: Processar imagem

Procedimento para execução:

1 - Selecionar uma imagem.

2 - Clicar no botão processar.

Resultado esperado: o sistema deve processar a imagem, apresentar uma nova imagem (processada); sem rastros e identificando os *spots,* versus a imagem original.

Resultado obtido: imagem processada corretamente

Sugestões de melhoria: melhorar o filtro de detecção dos *spots,* pois o algoritmo ainda pode ser melhorado

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 6

Sistema Operacional: Ubuntu *Desktop* 9.09

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: Processar imagem

Procedimento para execução:

1 - Selecionar uma imagem.
2 - Clicar no botão processar.

Resultado esperado: o sistema deve processar a imagem, apresentar uma nova imagem (processada); sem rastros e identificando os *spots, versus* a imagem original.

Resultado obtido: imagem processada corretamente

Sugestões de melhoria: melhorar o filtro de detecção dos *spots,* pois o algoritmo ainda pode ser melhorado

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 7

Sistema Operacional: OpenSolaris

Data da Realização: 01 de dezembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: Processar imagem

Procedimento para execução:

1 - Selecionar uma imagem.
2 - Clicar no botão processar.

Resultado esperado: o sistema deve processar a imagem, apresentar uma nova imagem (processada); sem rastros e identificando os *spots,* versus a imagem original.

Resultado obtido: imagem processada corretamente

Sugestões de melhoria: melhorar o filtro de detecção dos *spots,* pois o algoritmo ainda pode ser melhorado

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 8

Sistema Operacional: Windows XP

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: equipe

Versão: Sofia.jar versão 1.0

Descrição: Processar imagem

Procedimento para execução:

1 - Selecionar uma imagem.
2 - Clicar no botão processar.

Resultado esperado: o sistema deve processar a imagem, apresentar uma nova imagem (processada); sem rastros e identificando os *spots,* versus a imagem original.

Resultado obtido: imagem processada corretamente

Sugestões de melhoria: melhorar o filtro de detecção dos *spots,* pois o algoritmo ainda pode ser melhorado

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 9

Sistema Operacional: Windows 7

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: Adicionar um *spot* em uma imagem processada.

Procedimento para execução:

1 - Processar imagem.
2 - Selecionar na imagem original (à esquerda) o *spot* a ser adicionado.

Resultado esperado: sistema adiciona o *spot* à imagem já processada.

Resultado obtido: adição do *spot* efetuado corretamente.

Sugestões de melhoria: Corrigir os erros encontrados.

Defeitos não relacionados ao caso de teste: quando abre uma segunda imagem sem processá-la ele adiciona partes da imagem anterior na nova imagem.

Casos de Teste 10

Sistema Operacional: Ubuntu *Desktop* 9.09

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: Adicionar um *spot* em uma imagem processada.

Procedimento para execução:

1 - Processar imagem.
2 - Selecionar na imagem original (à esquerda) o *spot* a ser adicionado.

Resultado esperado: sistema adiciona o *spot* à imagem já processada.

Resultado obtido: adição do *spot* efetuado corretamente.

Sugestões de melhoria: Corrigir os erros encontrados.

Defeitos não relacionados ao caso de teste: quando abre uma segunda imagem sem processá-la ele adiciona partes da imagem anterior na nova imagem.

Caso de Teste 11

Sistema Operacional: OpenSolaris

Data da Realização: 01 de dezembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: Adicionar um *spot* em uma imagem processada.

Procedimento para execução:

1 - Processar imagem.
2 - Selecionar na imagem original (à esquerda) o *spot* a ser adicionado.

Resultado esperado: sistema adiciona o *spot* à imagem já processada.

Resultado obtido: adição do *spot* efetuado corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado

Caso de Teste 12

Sistema Operacional: Windows XP

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: equipe

Versão: Sofia.jar versão 1.0

Descrição: Adicionar um *spot* em uma imagem processada.

Procedimento para execução:

1 - Processar imagem.
2 - Selecionar na imagem original (à esquerda) o *spot* a ser adicionado.

Resultado esperado: sistema adiciona o *spot* à imagem já processada.

Resultado obtido: adição do *spot* efetuado corretamente.

Sugestões de melhoria: não há sugestões

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 13

Sistema Operacional: Windows 7

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: validar se a imagem processada manteve as mesmas propriedades através dos gráficos.

Procedimento para execução:

1 - Processar imagem.
2 - Clicar no botão gráfico.
3 - Selecionar uma seção da imagem.
4 - Sistema abre um *pop up* com o histograma da imagem e o trecho da imagem original e a nova imagem para ser comparado.

Resultado esperado: sistema deve carregar a tela de gráficos corretamente.

Resultado obtido: comparação entre as duas imagens através do histograma funciona corretamente.

Sugestões de melhoria: gerar um histograma em 3D.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 14

Sistema Operacional: Ubuntu *Desktop* 2009

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: validar se a imagem processada manteve as mesmas propriedades através dos gráficos.

Procedimento para execução:

1 - Processar imagem.
2 - Clicar no botão gráfico.
3 - Selecionar uma seção da imagem.
4 - Sistema abre um *pop up* com o histograma da imagem e o trecho da imagem original e a nova imagem para ser comparado.

Resultado esperado: sistema deve carregar a tela de gráficos corretamente.

Resultado obtido: comparação entre as duas imagens através do histograma funciona corretamente.

Sugestões de melhoria: gerar um histograma em 3D.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 15

Sistema Operacional: OpenSolaris

Data da Realização: 01 de dezembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: validar se a imagem processada manteve as mesmas propriedades através dos gráficos.

Procedimento para execução:

1 - Processar imagem.
2 - Clicar no botão gráfico.
3 - Selecionar uma seção da imagem.
4 - Sistema abre um *pop up* com o histograma da imagem e o trecho da imagem original e a nova imagem para ser comparado.

Resultado esperado: sistema deve carregar a tela de gráficos corretamente.

Resultado obtido: comparação entre as duas imagens através do histograma funciona corretamente.

Sugestões de melhoria: gerar um histograma em 3D.

Defeitos não relacionados ao caso de teste: nenhum.

Caso de Teste 16

Sistema Operacional: Windows XP

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: equipe

Versão: Sofia.jar versão 1.0

Descrição: validar se a imagem processada manteve as mesmas propriedades através dos gráficos.

Procedimento para execução:

1 - Processar imagem.
2 - Clicar no botão gráfico.
3 - Selecionar uma seção da imagem.

Resultado esperado: sistema deve carregar a tela de gráficos corretamente.

Resultado obtido: comparação entre as duas imagens através do histograma funciona corretamente.

Sugestões de melhoria: gerar um histograma em 3D.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 17

Sistema Operacional: Windows 7

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: trocar de imagem

Procedimento para execução:

1 - Clicar no botão abrir.
2 - Procurar nova imagem no computador.

Resultado esperado: sistema deve carregar a nova imagem para o *software*.

Resultado obtido: imagem trocada corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 18

Sistema Operacional: Ubuntu *Desktop* 9.09

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: trocar de imagem

Procedimento para execução:

1 - Clicar no botão abrir.
2 - Procurar nova imagem no computador.

Resultado esperado: sistema deve carregar a nova imagem para o *software*.

Resultado obtido: imagem trocada corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 19

Sistema Operacional: OpenSolaris

Data da Realização: 01 de dezembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: trocar de imagem

Procedimento para execução:

1 - Clicar no botão abrir.
2 - Procurar nova imagem no computador.

Resultado esperado: sistema deve carregar a nova imagem para o *software*.

Resultado obtido: imagem trocada corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 20

Sistema Operacional: Windows XP

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: equipe

Versão: Sofia.jar versão 1.0

Descrição: trocar de imagem

Procedimento para execução:

1 - Clicar no botão abrir.
2 - Procurar nova imagem no computador.

Resultado esperado: sistema deve carregar a nova imagem para o *software*.

Resultado obtido: imagem trocada corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 21

Sistema Operacional: Windows 7

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: formato inválido da imagem

Procedimento para execução:

1 - Clicar no botão abrir.
2 - Procurar nova imagem no computador.
3 - Imagem escolhida é em formato incompatível com o do sistema (diferente de JPEG, BITMAP ou TIFF).

Resultado esperado: sistema só deve mostrar os formatos compatíveis.

Resultado obtido: imagens que podem ser abertas pelo *software* visualizadas corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 22

Sistema Operacional: Ubuntu *Desktop* 9.09

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: formato inválido da imagem

Procedimento para execução:

1 - Clicar no botão abrir.
2 - Procurar nova imagem no computador.
3 - Imagem escolhida é em formato incompatível com o do sistema (diferente de JPEG, BITMAP ou TIFF).

Resultado esperado: sistema só deve mostrar os formatos compatíveis.

Resultado obtido: imagens que podem ser abertas pelo *software* visualizadas corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 23

Sistema Operacional: OpenSolaris

Data da Realização: 01 de dezembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: formato inválido da imagem

Procedimento para execução:

1 - Clicar no botão abrir.
2 - Procurar nova imagem no computador.
3 - Imagem escolhida é em formato incompatível com o do sistema (diferente de JPEG, BITMAP ou TIFF).

Resultado esperado: sistema só deve mostrar os formatos compatíveis.

Resultado obtido: imagens que podem ser abertas pelo *software* visualizadas corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 24

Sistema Operacional: Windows XP

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: equipe

Versão: Sofia.jar versão 1.0

Descrição: formato inválido da imagem

Procedimento para execução:

1 - Clicar no botão abrir.
2 - Procurar nova imagem no computador.
3 - Imagem escolhida é em formato incompatível com o do sistema (diferente de JPEG, BITMAP ou TIFF).

Resultado esperado: sistema deve avisar “Favor selecionar formato válido” e não carregar a imagem.

Resultado obtido: mensagem de erro apresentada.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 25

Sistema Operacional: Windows 7

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: excluir detalhes

Procedimento para execução:

1 - Selecionar na imagem processada o *spot* a ser excluído.

Resultado esperado: sistema deve excluir o *spot* na imagem processada.

Resultado obtido: *spot* excluído corretamente

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 26

Sistema Operacional: Ubuntu *Desktop* 9.09

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: excluir detalhes

Procedimento para execução:

1 - Selecionar na imagem processada o *spot* a ser excluído.

Resultado esperado: sistema deve excluir o *spot* na imagem processada.

Resultado obtido: *spot* excluído corretamente

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum

Caso de Teste 27

Sistema Operacional: OpenSolaris

Data da Realização: 01 de dezembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: excluir detalhes

Procedimento para execução:

1 - Selecionar na imagem processada o *spot* a ser excluído.

Resultado esperado: sistema deve excluir o *spot* na imagem processada.

Resultado obtido: *spot* excluído corretamente

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 28

Sistema Operacional: Windows XP

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: equipe

Versão: Sofia.jar versão 1.0

Descrição: excluir detalhes

Procedimento para execução:

1 - Selecionar na imagem processada o *spot* a ser excluído.

Resultado esperado: sistema deve excluir o *spot* na imagem processada.

Resultado obtido: *spot* excluído corretamente.

Sugestões de melhoria: não há sugestões.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 29

Sistema Operacional: Windows 7

Data da Realização: 30 de novembro de 2009

Versão: Sofia.jar versão 1.0

Descrição: erro de processamento

Procedimento para execução:

1 - Selecionar uma imagem sem estar na escala de cinza.

Resultado esperado: sistema emite mensagem: “Favor selecionar outra imagem compatível”.

Resultado obtido: todas as imagens coloridas ou não são carregadas

Sugestões de melhoria: por a imagem do gel 2D já vir com escala de cinza provavelmente não afetará no caso de teste, e caso alguma imagem seja carregada colorida também não afetará no desempenho.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 30

Sistema Operacional: Ubuntu *Desktop* 9.09

Data da Realização: 30 de novembro de 2009

Versão: Sofia.jar versão 1.0

Descrição: erro de processamento

Procedimento para execução:

1 - Selecionar uma imagem sem estar na escala de cinza.

Resultado esperado: sistema emite mensagem: “Favor selecionar outra imagem compatível”.

Resultado obtido: todas as imagens coloridas ou não são carregadas

Sugestões de melhoria: por a imagem do gel 2D já vir com escala de cinza provavelmente não afetará no caso de teste, e caso alguma imagem seja carregada colorida também não afetará no desempenho.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 31

Sistema Operacional: OpenSolaris

Data da Realização: 01 de dezembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: erro de processamento

Procedimento para execução:

1 - Selecionar uma imagem sem estar na escala de cinza.

Resultado esperado: sistema emite mensagem: “Favor selecionar outra imagem compatível”.

Resultado obtido: todas as imagens coloridas ou não são carregadas

Sugestões de melhoria: por a imagem do gel 2D já vir com escala de cinza provavelmente não afetará no caso de teste, e caso alguma imagem seja carregada colorida também não afetará no desempenho.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

Caso de Teste 32

Sistema Operacional: Windows XP

Data da Realização: 30 de novembro de 2009

Responsável pelo teste: Juliana

Versão: Sofia.jar versão 1.0

Descrição: erro de processamento

Procedimento para execução:

1 - Selecionar uma imagem sem estar na escala de cinza.

Resultado esperado: sistema emite mensagem: “Favor selecionar outra imagem compatível”.

Resultado obtido: todas as imagens coloridas ou não são carregadas

Sugestões de melhoria: por a imagem do gel 2D já vir com escala de cinza provavelmente não afetará no caso de teste, e caso alguma imagem seja carregada colorida também não afetará no desempenho.

Defeitos não relacionados ao caso de teste: nenhum defeito encontrado.

RESUMO DOS CASOS DE TESTES

A fim de facilitar a compreensão e visualização dos testes ao decorrer do projeto, foi elaborada uma tabela com os resultados obtidos em cada uma das versões.

Resumo dos Casos de Teste					
		resultado obtido por sistema operacional			
Descrição	versão	Win XP	Win 7	Ubuntu 0.09	OpenSolaris
carregar imagem					
	versão 0.1	ok			
	versão 0.4	ok	ok	ok	ok
	versão 1.0		ok	ok	ok
processar imagem					
	versão 0.1	ok			
	versão 0.4	ok	ok	ok	ok
	versão 1.0	ok	ok	ok	ok
Adicionar um *spot* em uma imagem processada					
	versão 0.1	ok			
	versão 0.4	ok	ok	ok	ok
	versão 1.0		ok	ok	
validar a imagem se a imagem processada manteve as mesmas propriedades através dos gráficos					
	versão 0.1				
	versão 0.4				
	versão 1.0	ok	ok	ok	ok
trocar de imagem					
	versão 0.1	ok			
	versão 0.4	ok	ok	ok	ok
	versão 1.0	ok	ok	ok	ok
formato inválido da imagem					
	versão 0.1				
	versão 0.4				
	versão 1.0	ok	ok	ok	ok
excluir detalhes					
	versão 0.1				
	versão 0.4	ok	ok	ok	ok
	versão 1.0	ok	ok	ok	ok
erro de processamento					
	versão 0.1	ok			
	versão 0.4	ok	ok	ok	ok
	versão 1.0	ok	ok	ok	ok
salvar imagem					
	versão 0.1	ok			
	versão 0.4	ok	ok	ok	ok
	versão 1.0	ok	ok	ok	ok

Tabela 2 - Resumo dos casos de teste
Fonte: Trabalho de Conclusão de Curso

APÊNDICE 3 – DIAGRAMAS DE CLASSES

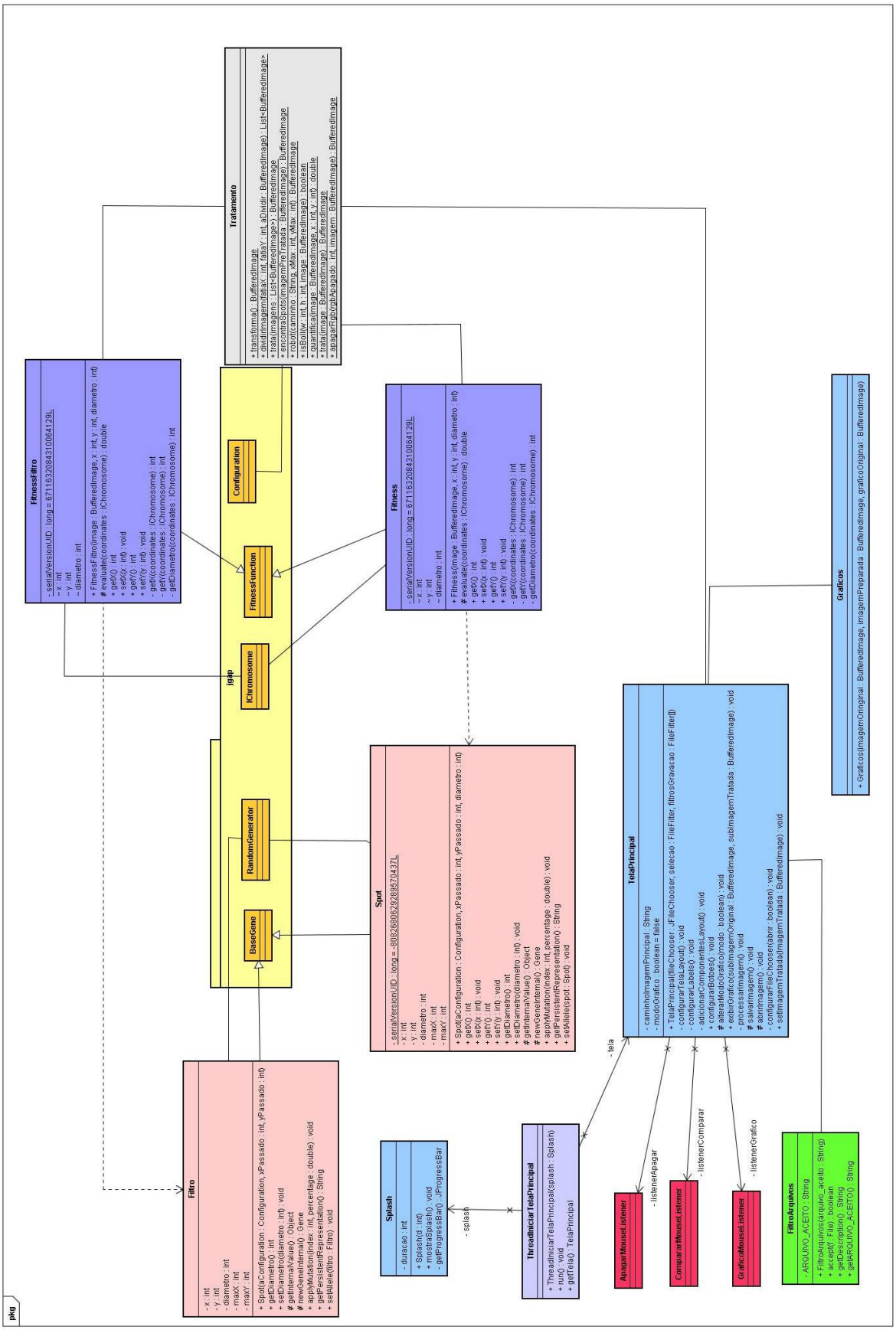


Figura 23 - Diagrama de classes
Fonte: Trabalho de Conclusão de curso

APÊNDICE 4 – DIAGRAMAS DE CASO DE USO

DIAGRAMA DE CASOS DE USO

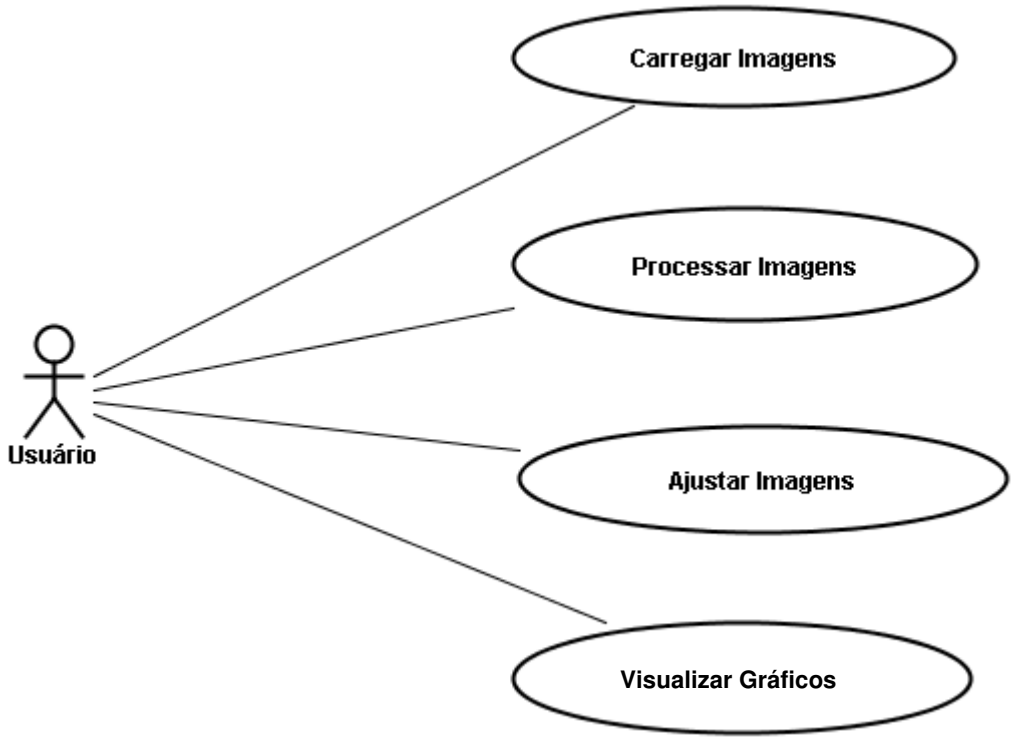


Figura 24 - Casos de uso
Fonte – Trabalho de Conclusão de Curso

UC001 – CARREGAR IMAGENS

Descrição

Este caso de uso descreve como carregar uma imagem para processamento.

Pré-condições

Este Caso de Uso pode iniciar somente se:

1. As imagens estiverem salvas no computador e/ou outro dispositivo.

Pós-condições

Após o fim normal deste Caso de Uso o sistema deve:

1. Ter realizado o carregamento da imagem no sistema.

Ator Primário

Usuário.

Fluxo de Eventos Principal

1. Usuário abre o sistema. (DV1)
2. Usuário clica no botão Abrir. (DV2)
3. Usuário procura a nova imagem no computador. (E1)
4. Sistema carrega a imagem. (DV3) (E2) (R1)
5. O caso de uso é finalizado.

Fluxos de Exceção

E1: Usuário deseja trocar imagem.

1. Usuário clica novamente no botão Abrir.
2. Usuário procura a nova imagem no computador.
3. Sistema carrega a imagem. (E2)
4. O caso de uso é finalizado.

E2: Formato inválido.

1. Sistema consiste que o formato do arquivo não é compatível.
2. Sistema emite a mensagem: “Favor selecionar formato válido”.
3. O caso de uso é finalizado.

Regras de Negócio

R1: Somente devem ser carregados arquivos de imagem. Formatos JPEG, BITMAP e TIFF.

Data View

DV1

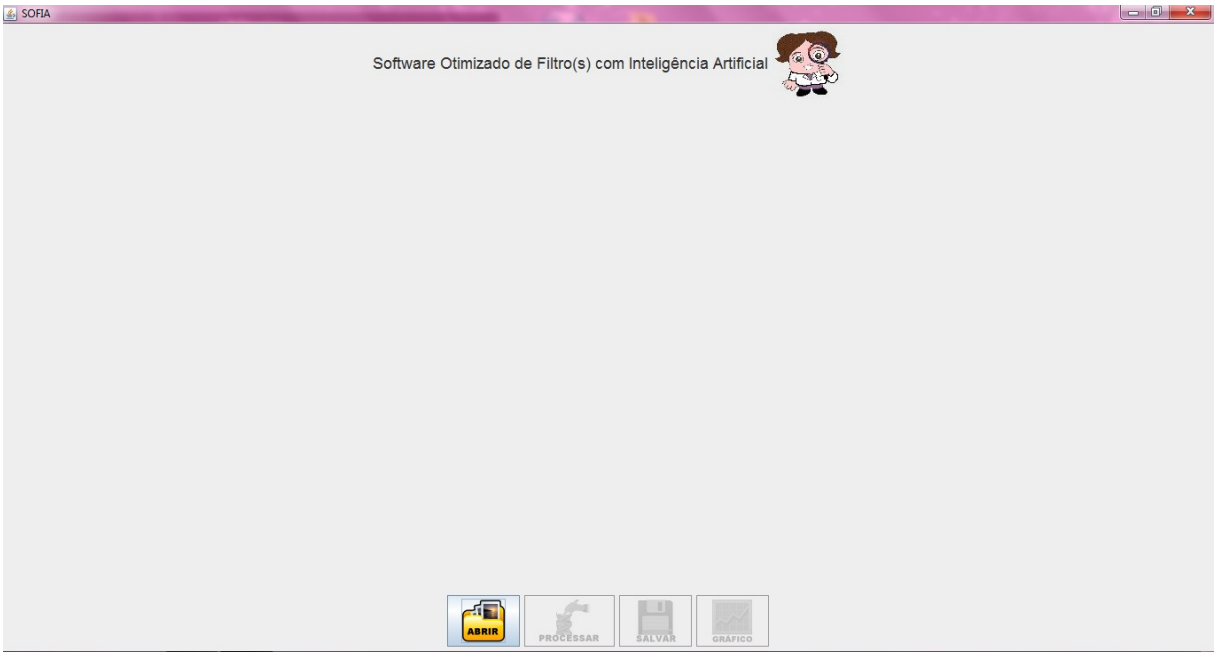


Figura 25 – Abertura do sistema
Fonte – Trabalho de Conclusão de Curso

DV2

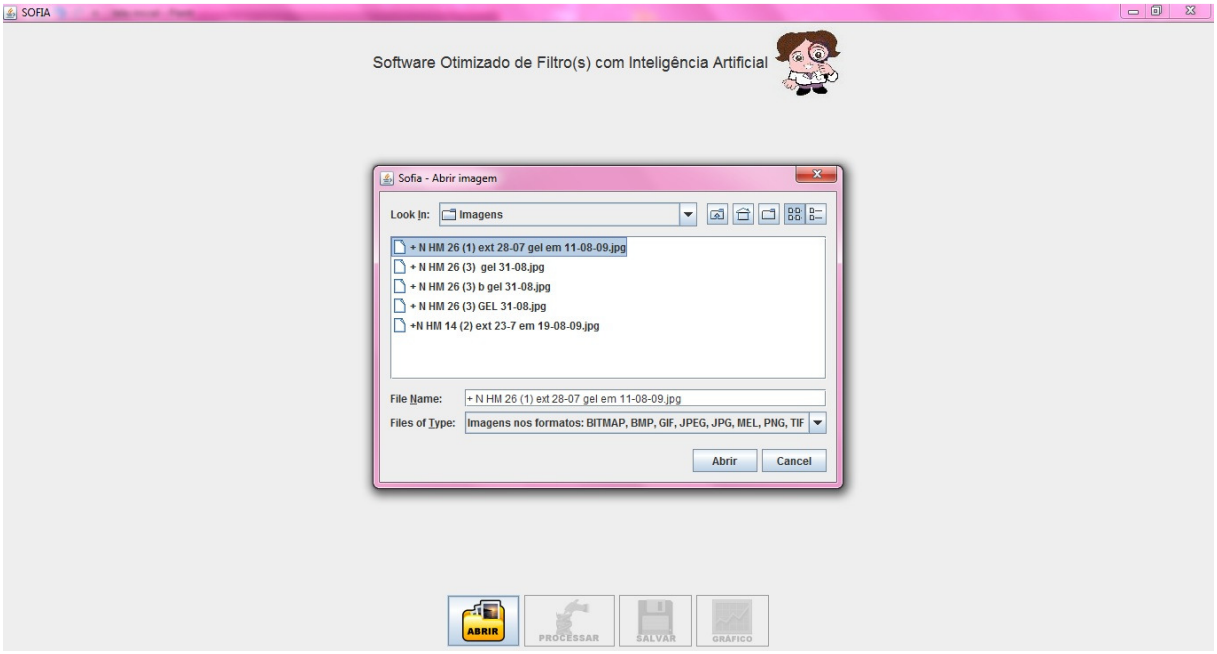


Figura 26 – Selecionar imagem
Fonte – Trabalho de Conclusão de Curso

DV3

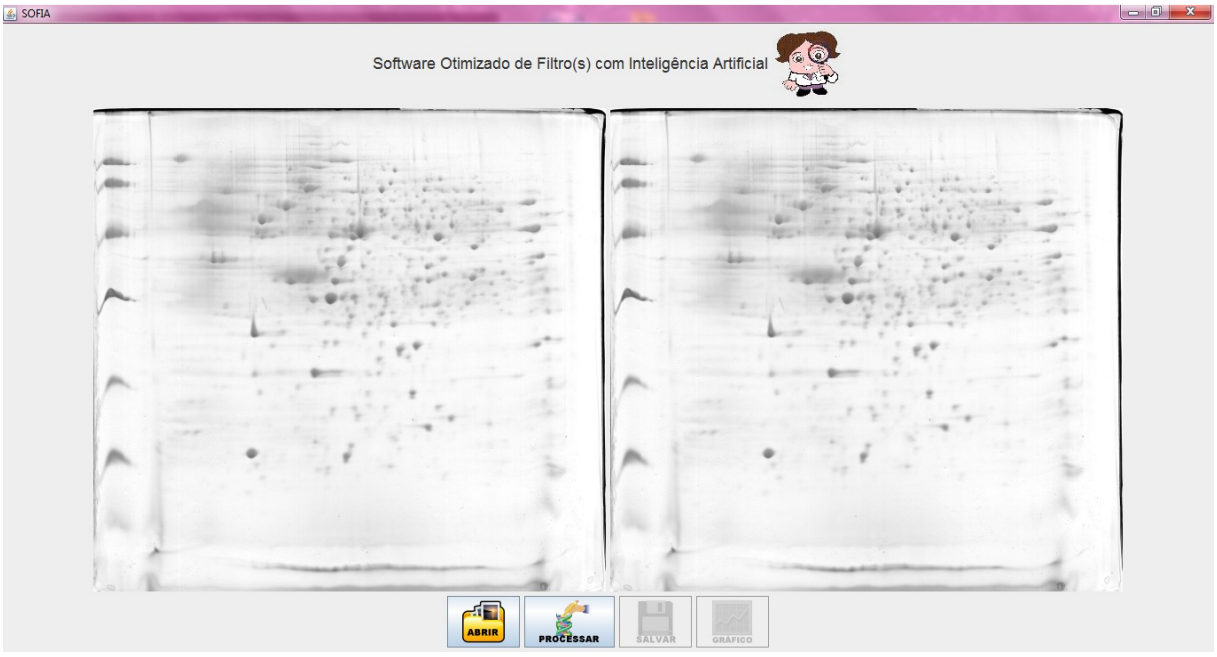


Figura 27 – Carregar imagem
Fonte – Trabalho de Conclusão de Curso

UC002 – PROCESSAR IMAGENS

Descrição

Este caso de uso descreve como processar uma imagem.

Pré-condições

Este Caso de Uso pode iniciar somente se:

1. As imagens devem ter sido previamente carregadas no sistema. (UC001)

Pós-condições

Após o fim normal deste Caso de Uso o sistema deve:

1. Ter realizado o processamento da imagem no sistema.

Ator Primário

Usuário.

Fluxo de Eventos Principal

1. Usuário carrega a imagem. (DV1)
2. Usuário clica no botão Processar.
3. Sistema processa a imagem. (DV2) (E1)
4. O caso de uso é finalizado.

Fluxos de Exceção

E1: Erro de processamento.

1. Sistema verifica se a imagem não se encontra em escala de cinza.
2. Sistema emite a mensagem: “Favor selecionar outra imagem compatível”.
3. O caso de uso é finalizado.

Data View

DV1

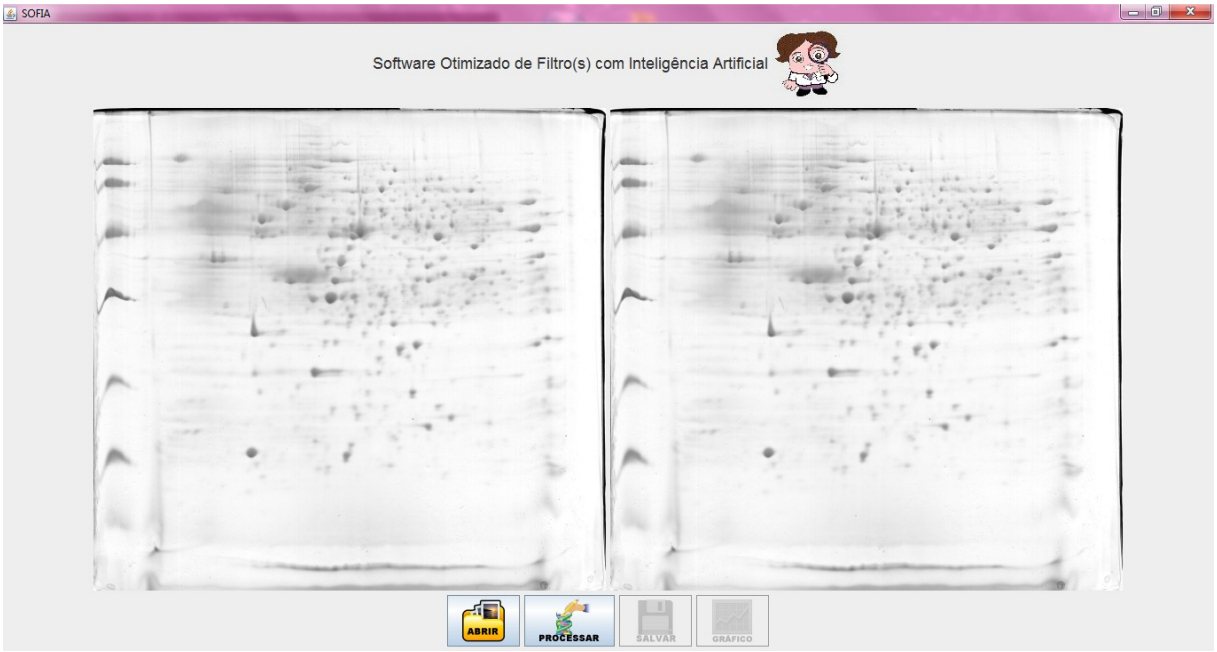


Figura 28 - Carregar imagem
Fonte – Trabalho de Conclusão de Curso

DV2

Figura 29 - Processar imagem
Fonte – Trabalho de Conclusão de Curso

UC003 – SELECIONAR *SPOTS*

Descrição

Este caso de uso descreve como adicionar um *spot* a uma imagem já processada.

Pré-condições

Este Caso de Uso pode iniciar somente se:

1. As imagens tiverem ter sido previamente processadas no sistema.

Pós-condições

Após o fim normal deste Caso de Uso o sistema deve:

1. Permitir ao usuário ter adicionado s*pots* à imagem.

Ator Primário

Usuário.

Fluxo de Eventos Principal

1. Usuário faz o processamento da imagem (UC002) (DV1)
2. Usuário seleciona na imagem original (à esquerda) o s*pot* a ser adicionado. (DV2)
3. Sistema adiciona o s*pot* à imagem já processada.
4. O caso de uso é finalizado.

Fluxos de Exceção

E1: Usuário deseja excluir detalhes.

1. Usuário seleciona na imagem processada o s*pot* a ser excluído.
2. Sistema exclui o s*pot* na imagem processada.
3. O caso de uso é finalizado.

Data View

DV1

Figura 30 - Processar imagem
Fonte – Trabalho de Conclusão de Curso

DV2

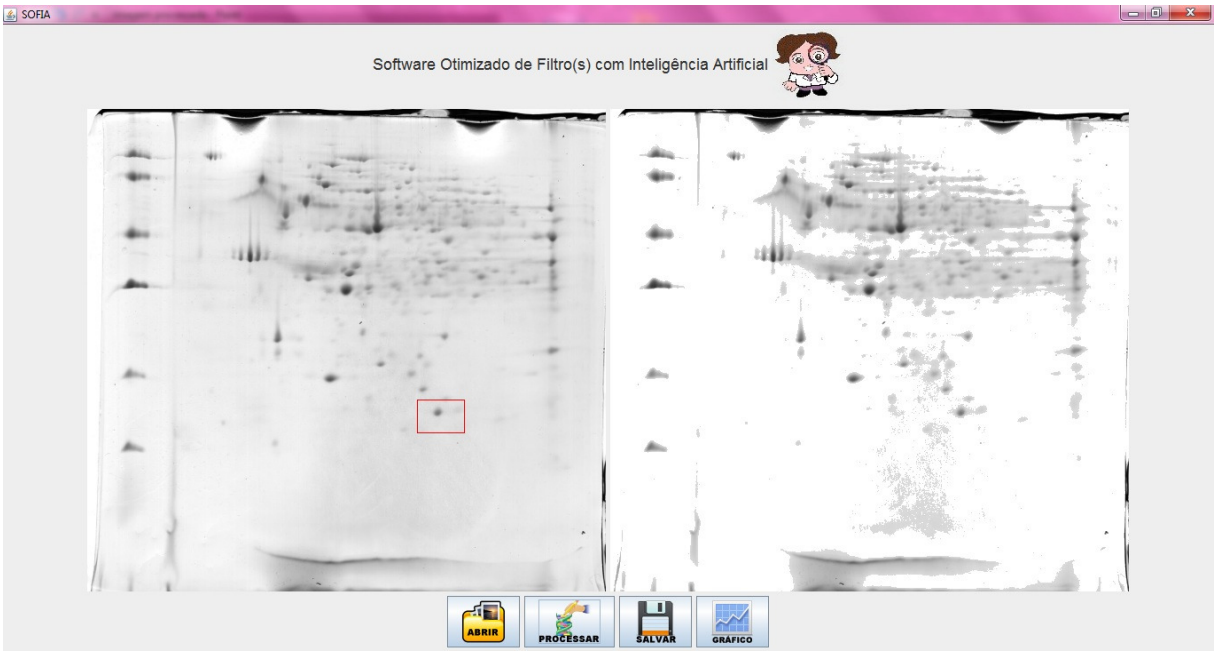


Figura 31 - Selecionar s*pot*
Fonte – Trabalho de Conclusão de Curso

UC004 – VISUALIZAR GRÁFICOS

Descrição

Este caso de uso descreve como validar se a imagem processada manteve as mesmas propriedades através dos gráficos.

Pré-condições

Este Caso de Uso pode iniciar somente se:

1. As imagens devem ter sido previamente processadas no sistema. (UC002)

Pós-condições

Após o fim normal deste Caso de Uso o sistema deve:

1. Permitir ao usuário ter visualizado os gráficos referentes a um a seção na imagem.

Ator Primário

Usuário.

Fluxo de Eventos Principal

1. Usuário faz o processamento da imagem (UC002) (DV1)
2. Usuário clica no botão "Gráficos".
3. Usuário seleciona uma seção da imagem. (DV2)
4. Sistema carrega a tela de gráficos conforme a (DV3)
5. O caso de uso é finalizado.

Data View

DV1

Figura 32 - Processar imagem
Fonte – Trabalho de Conclusão de Curso

DV2

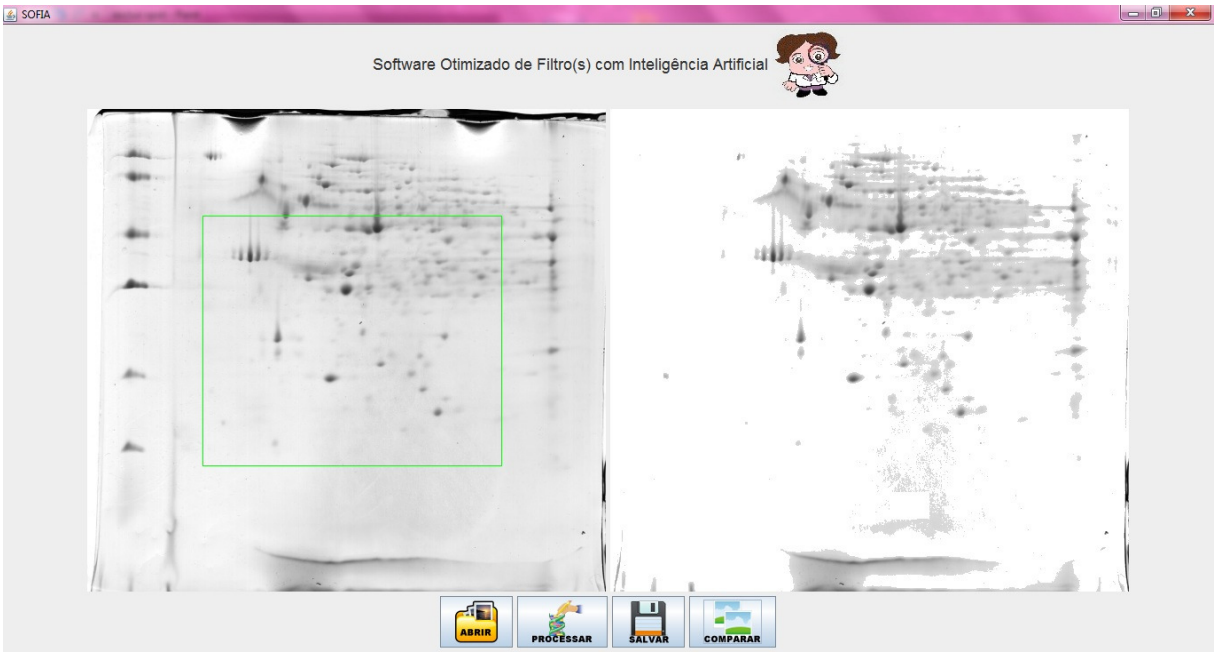


Figura 33 - Selecionar área do gráfico
Fonte – Trabalho de Conclusão de Curso

DV3

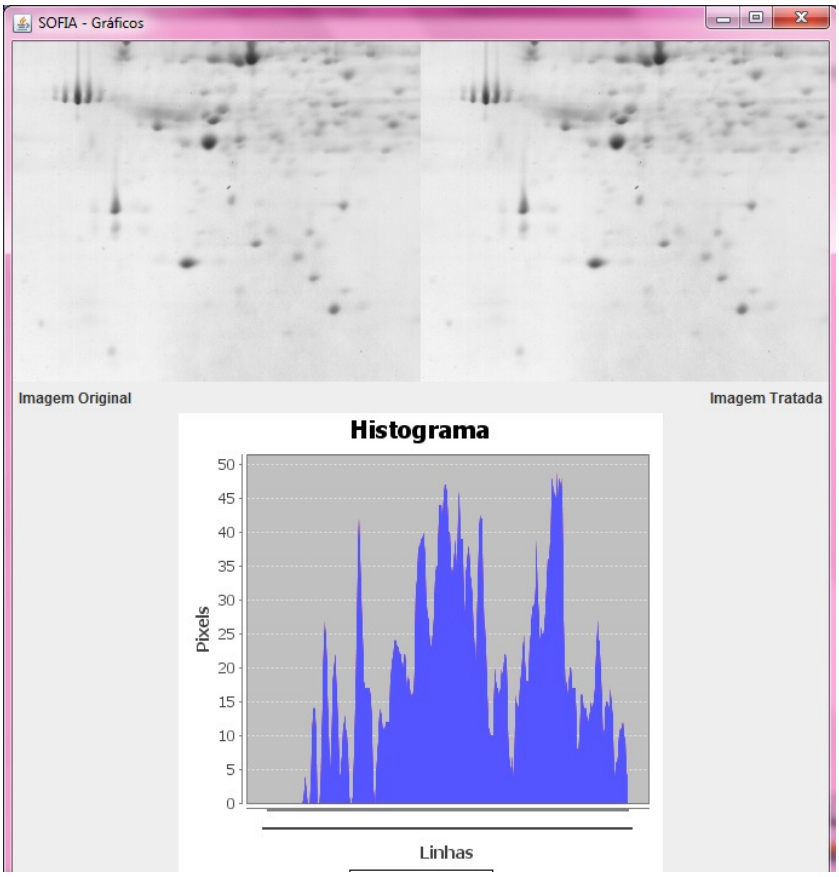


Figura 34 – Gráfico
Fonte – Trabalho de Conclusão de Curso

APÊNDICE 5 – DIAGRAMAS DE SEQÜÊNCIA

ABRIR IMAGEM

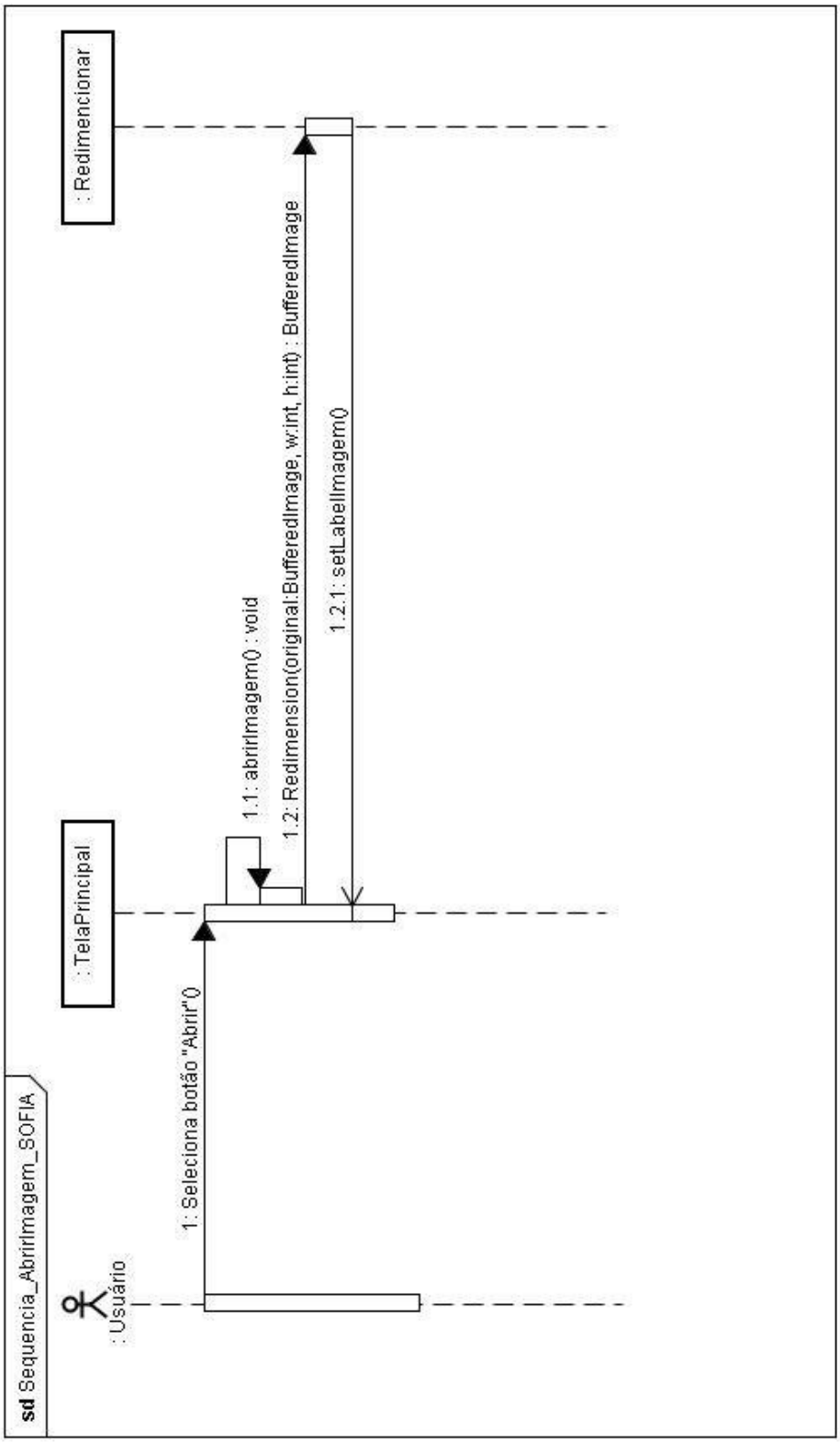


Figura 35 - Seqüência - Abrir imagem
Fonte – Trabalho de Conclusão de Curso

PROCESSAR IMAGEM

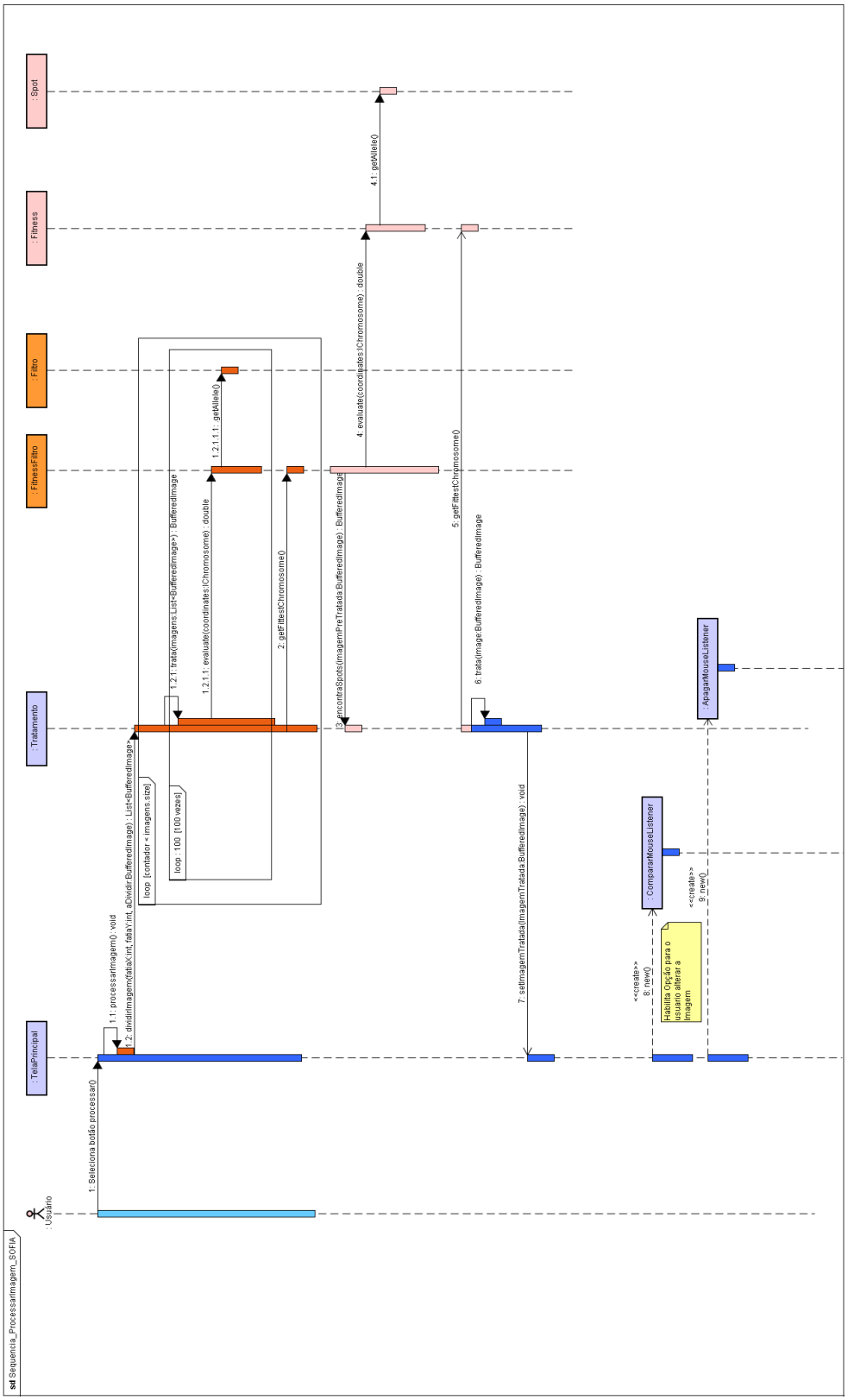


Figura 36 - Seqüência - Processar imagem
Fonte – Trabalho de Conclusão de Curso

SALVAR IMAGEM

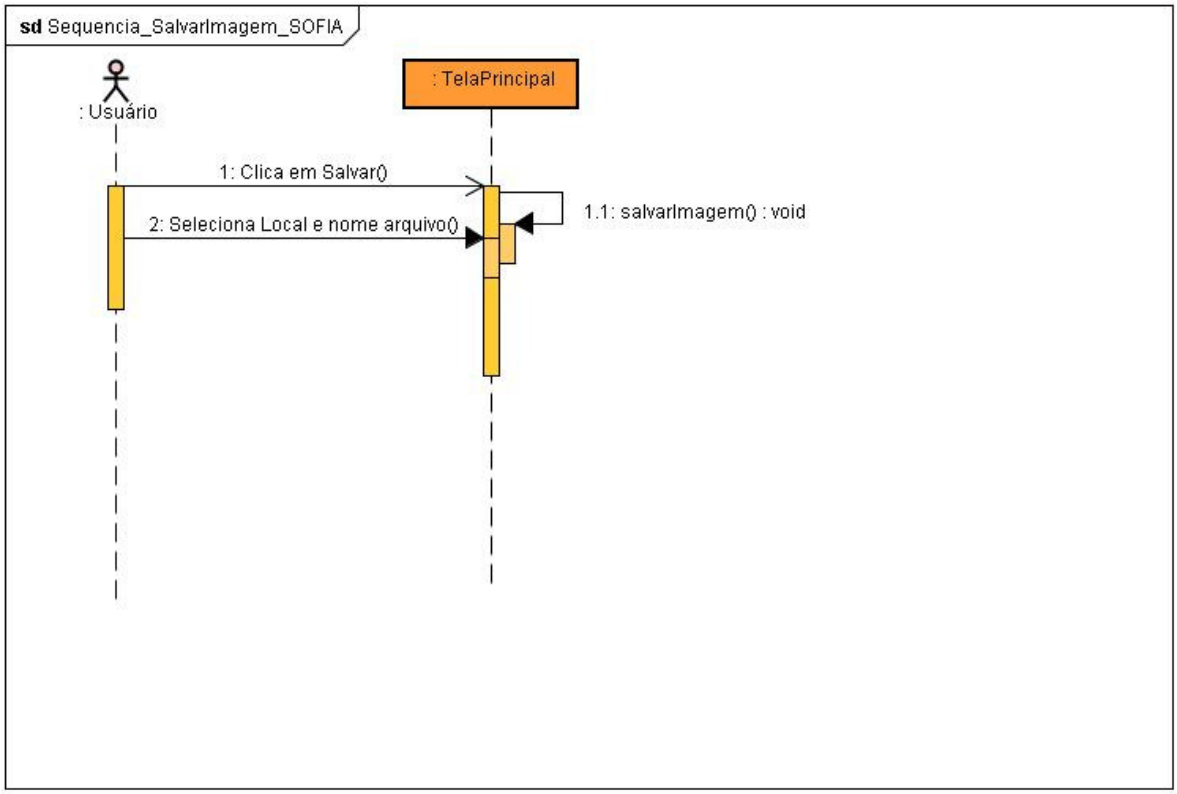


Figura 37 - Seqüência - Salvar imagem
Fonte – Trabalho de Conclusão de Curso

VISUALIZAR GRÁFICOS

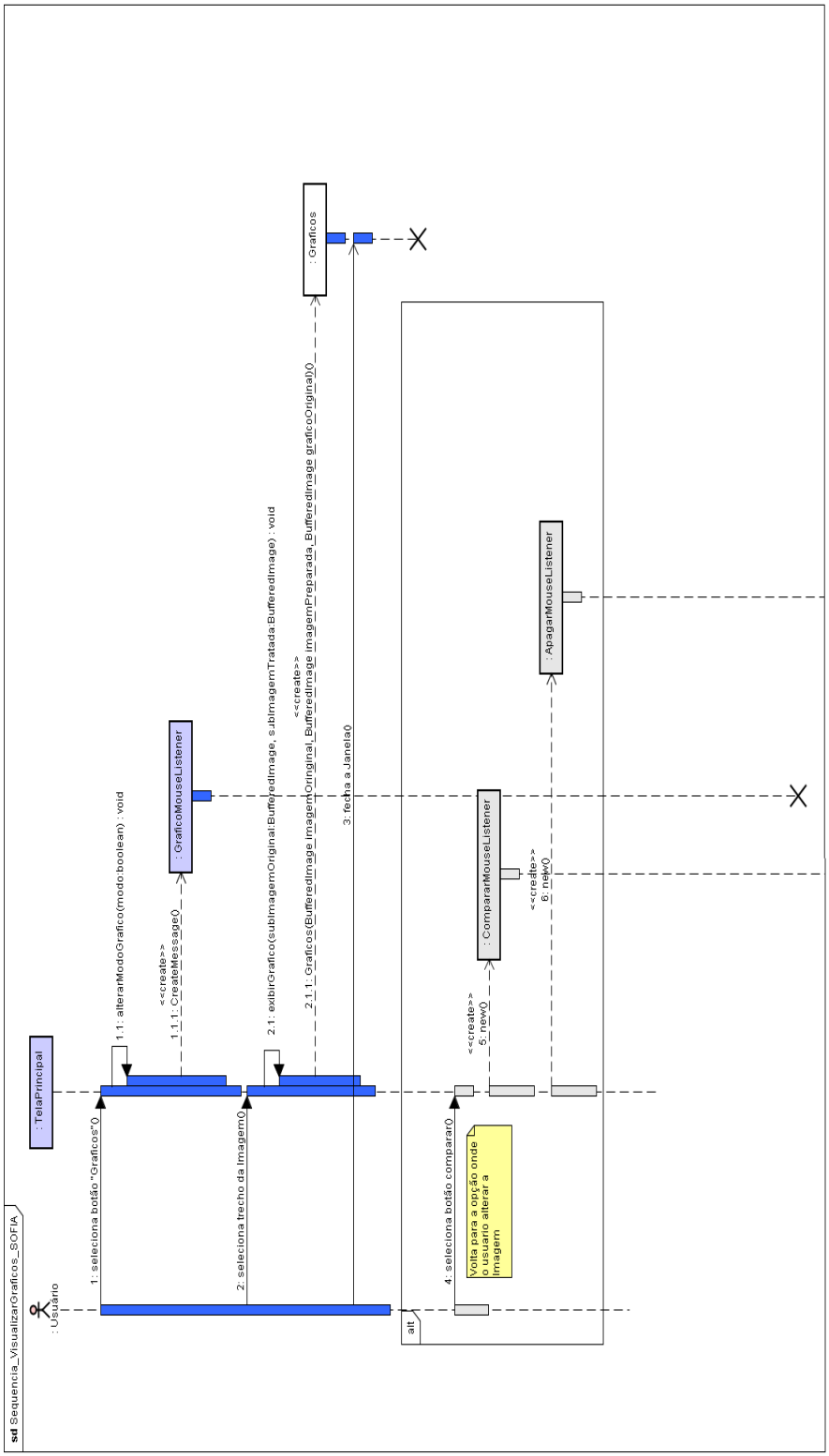


Figura 38 - Seqüência - Visualizar gráficos
Fonte – Trabalho de Conclusão de Curso

APÊNDICE 6 – DIAGRAMA DE COLABORAÇÃO

ABRIR IMAGEM

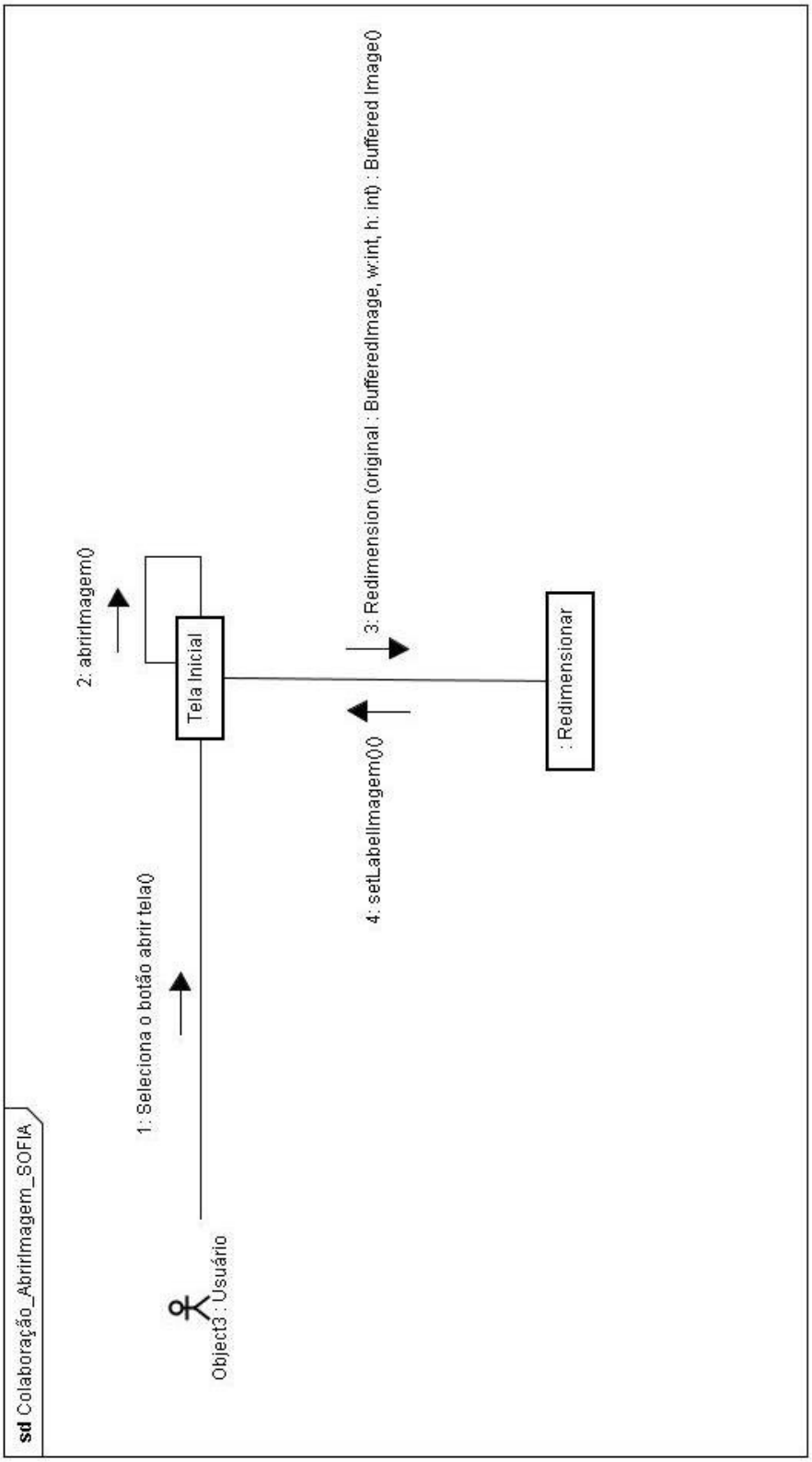


Figura 39 - Colaboração - Abrir imagem

Fonte – Trabalho de Conclusão de Curso

PROCESSAR IMAGEM

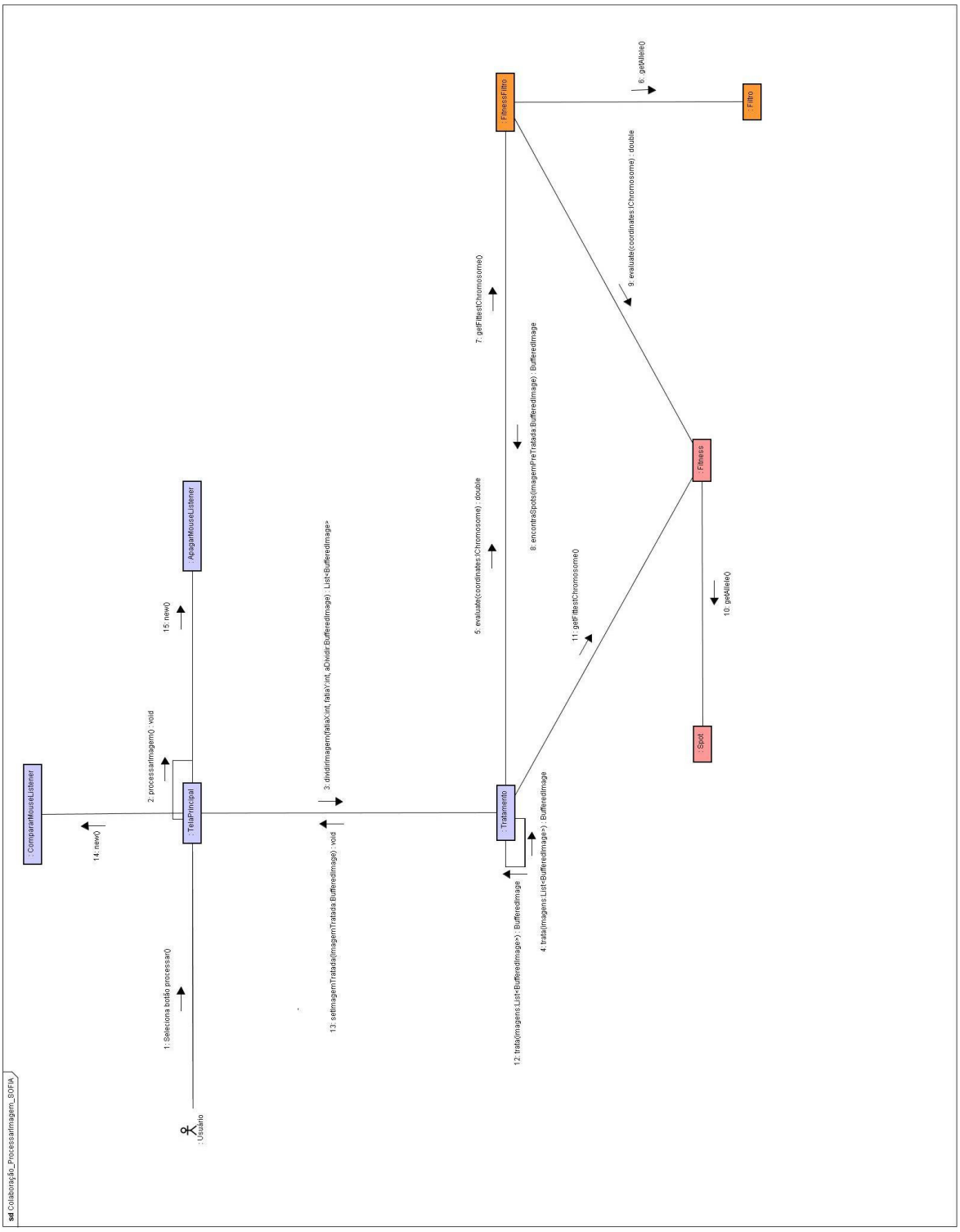


Figura 40 - Colaboração - Processar imagem
Fonte – Trabalho de Conclusão de Curso

SALVAR IMAGEM

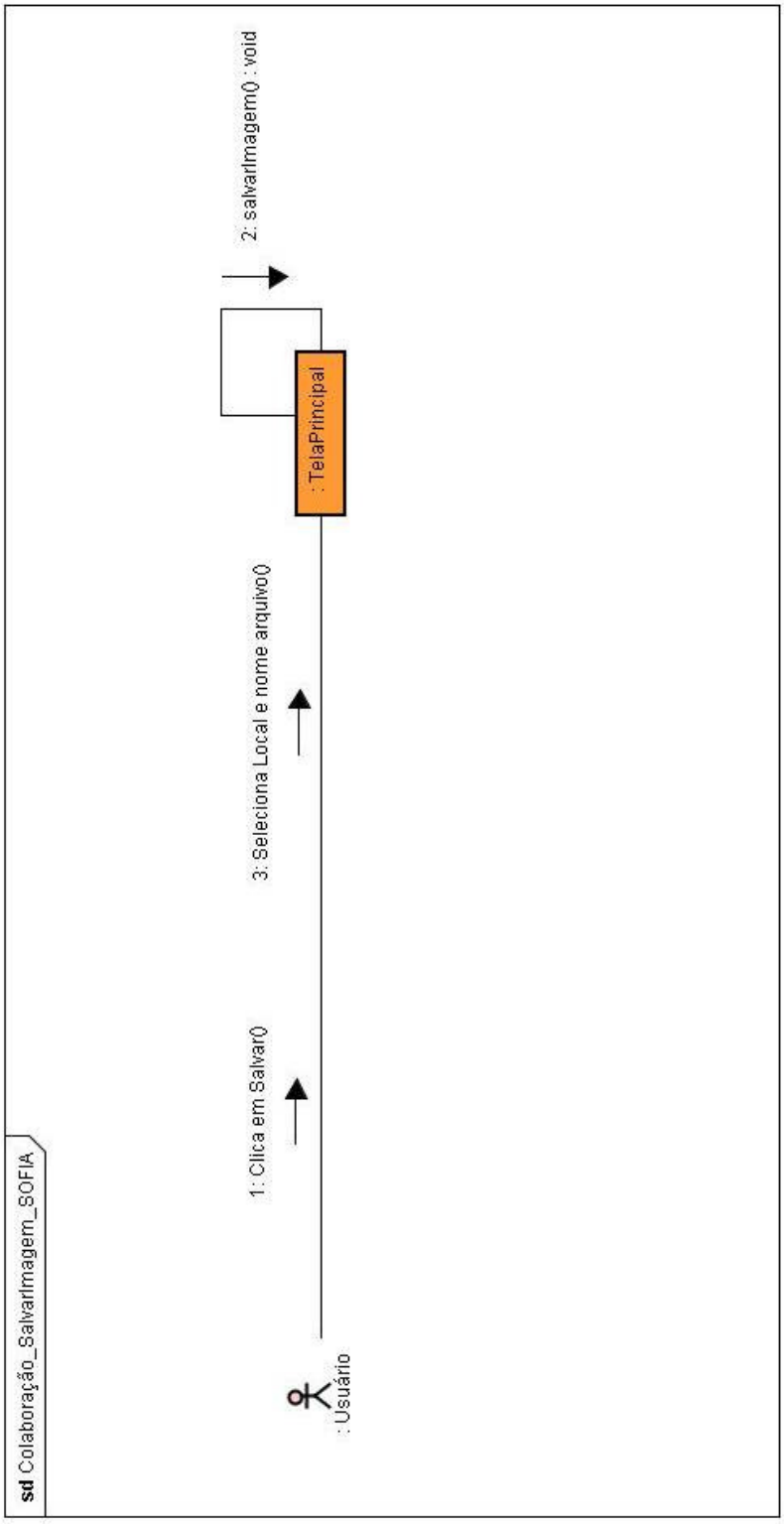


Figura 41 - Colaboração - Salvar imagem
Fonte – Trabalho de Conclusão de Curso

VISUALIZAR GRÁFICOS

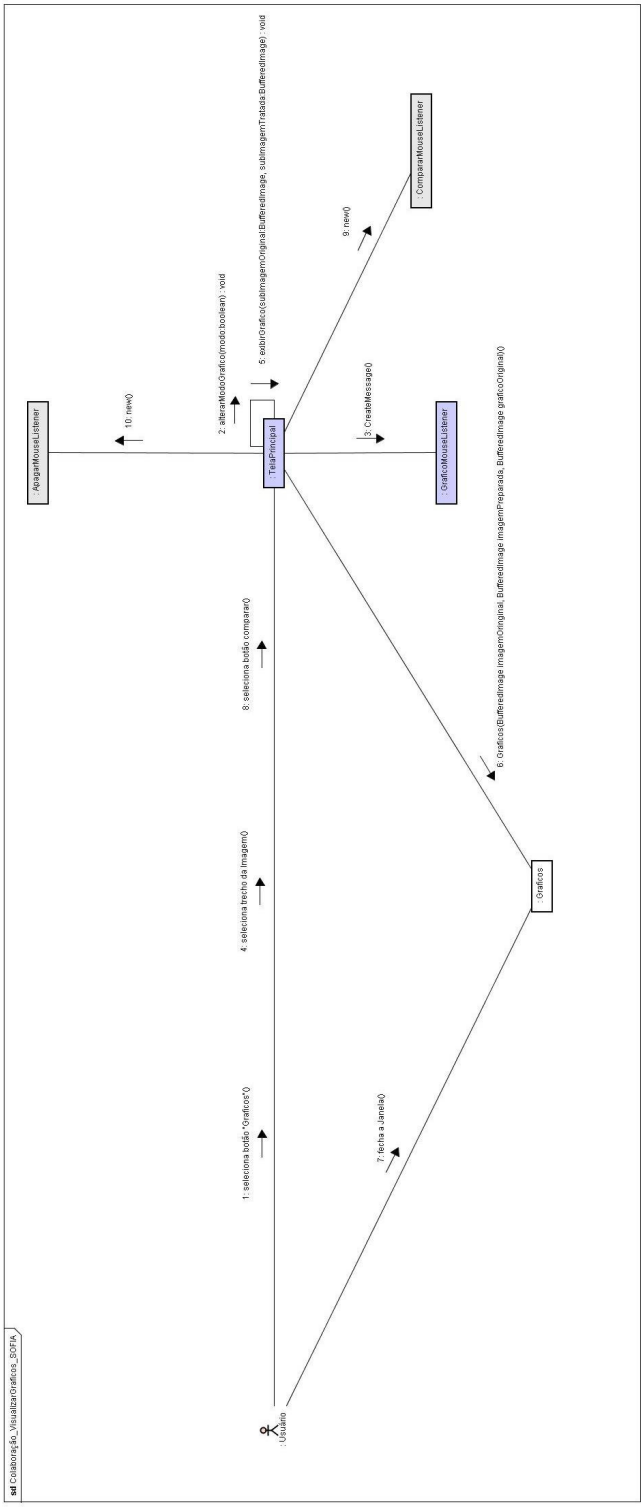


Figura 42 - Colaboração - Visualizar gráficos
Fonte – Trabalho de Conclusão de Curso

APÊNDICE 7 – DIAGRAMA DE ATIVIDADES

CARREGAR IMAGEM

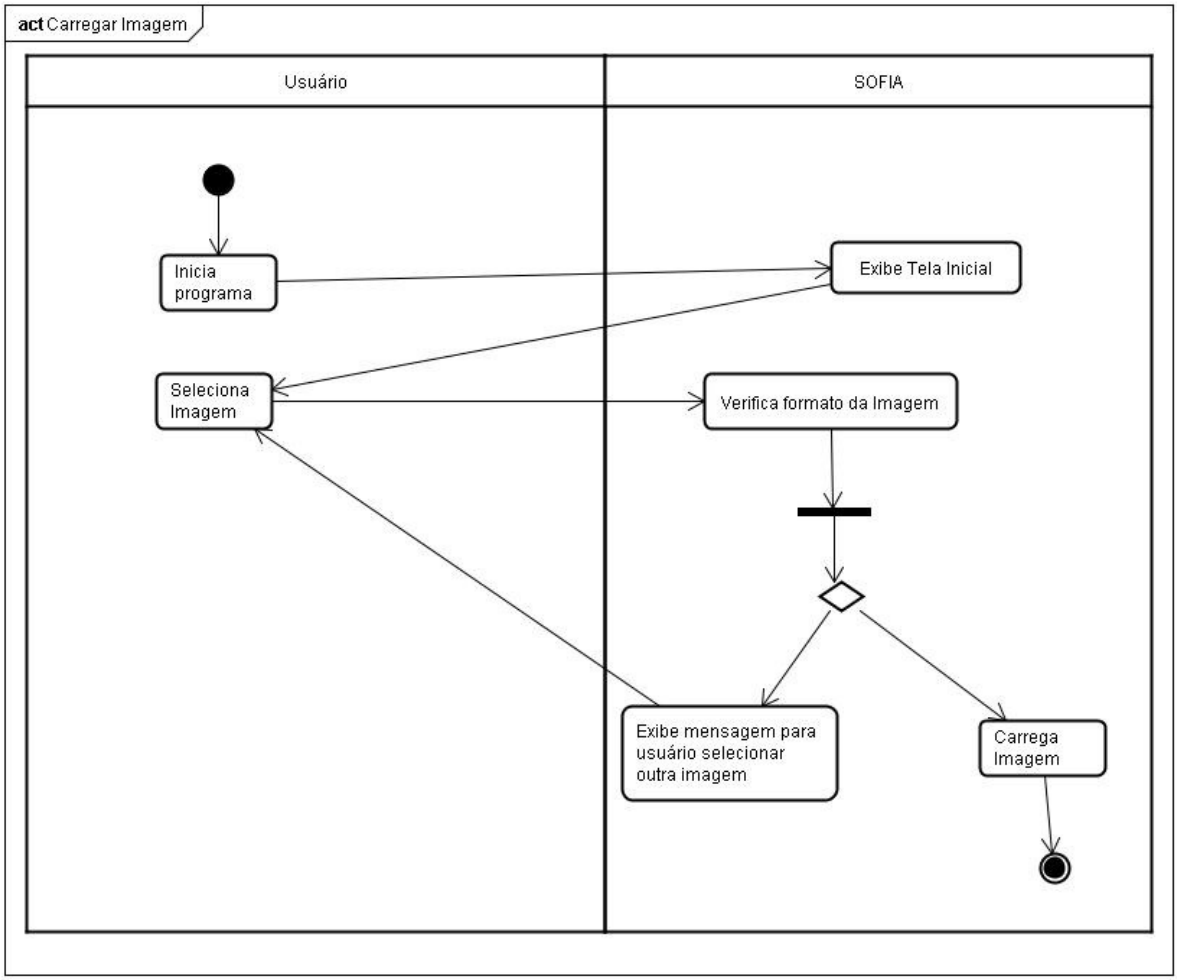


Figura 43 - Atividades - Carregar imagem
Fonte – Trabalho de Conclusão de Curso

PROCESSAR IMAGEM

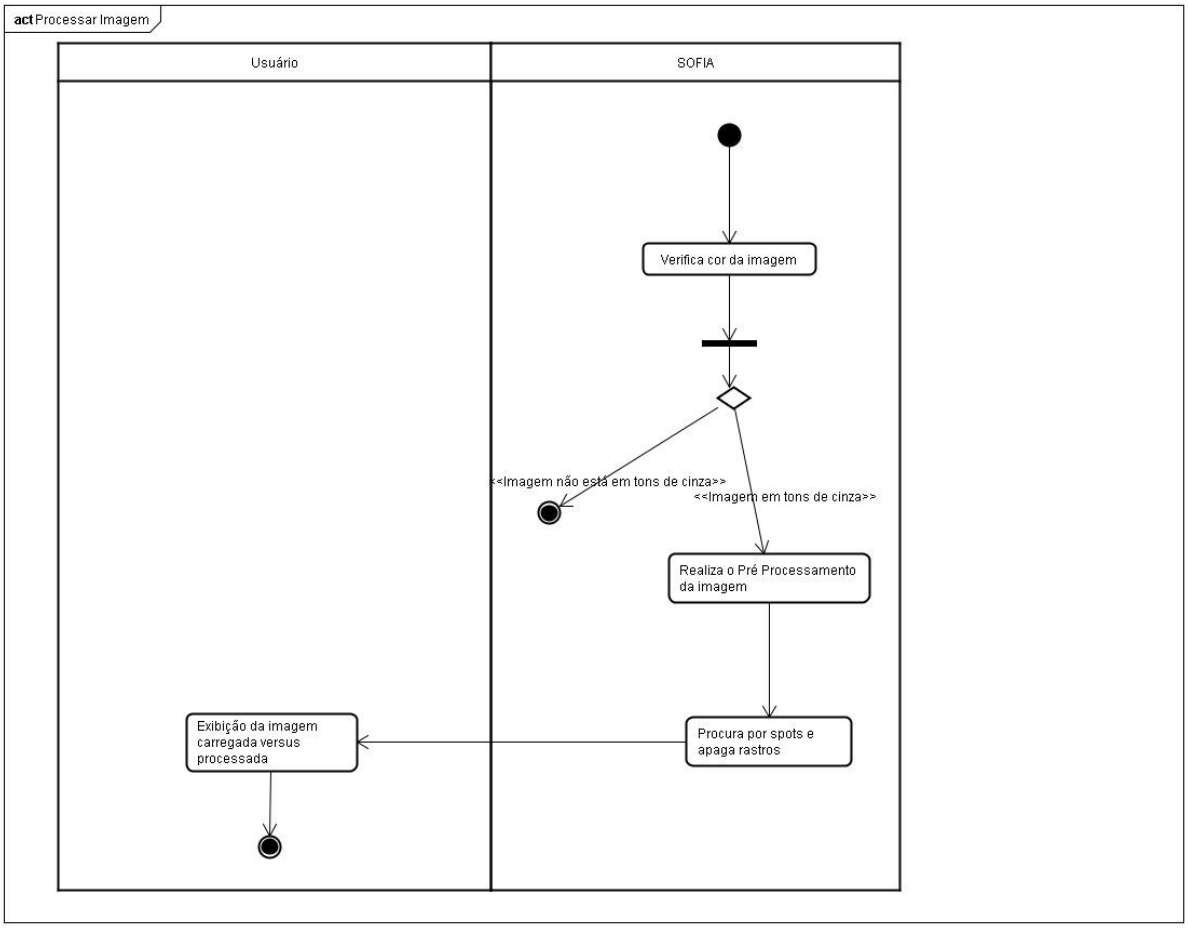


Figura 44 - Atividades - Processar imagem
Fonte – Trabalho de Conclusão de Curso

SELECIONAR *SPOTS*

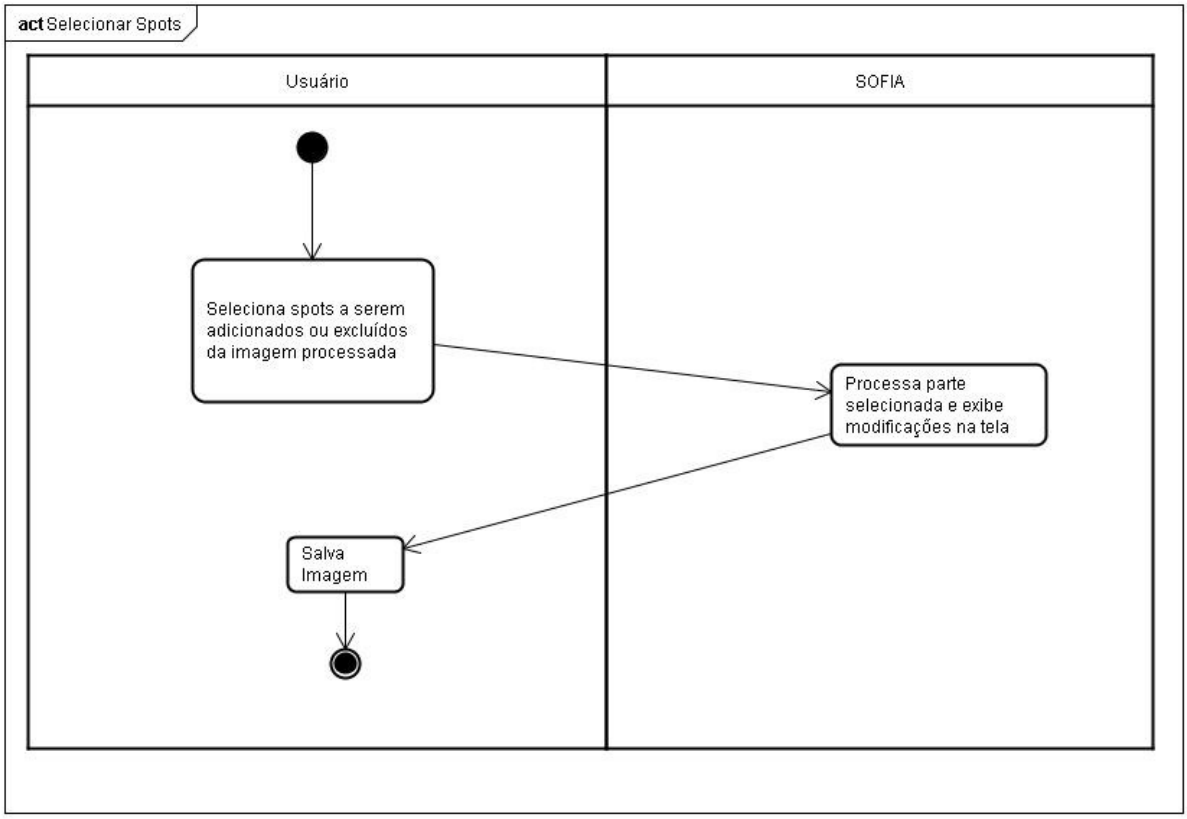


Figura 45 - Atividades - Selecionar s*pots*
Fonte – Trabalho de Conclusão de Curso

VISUALIZAR GRAFICOS

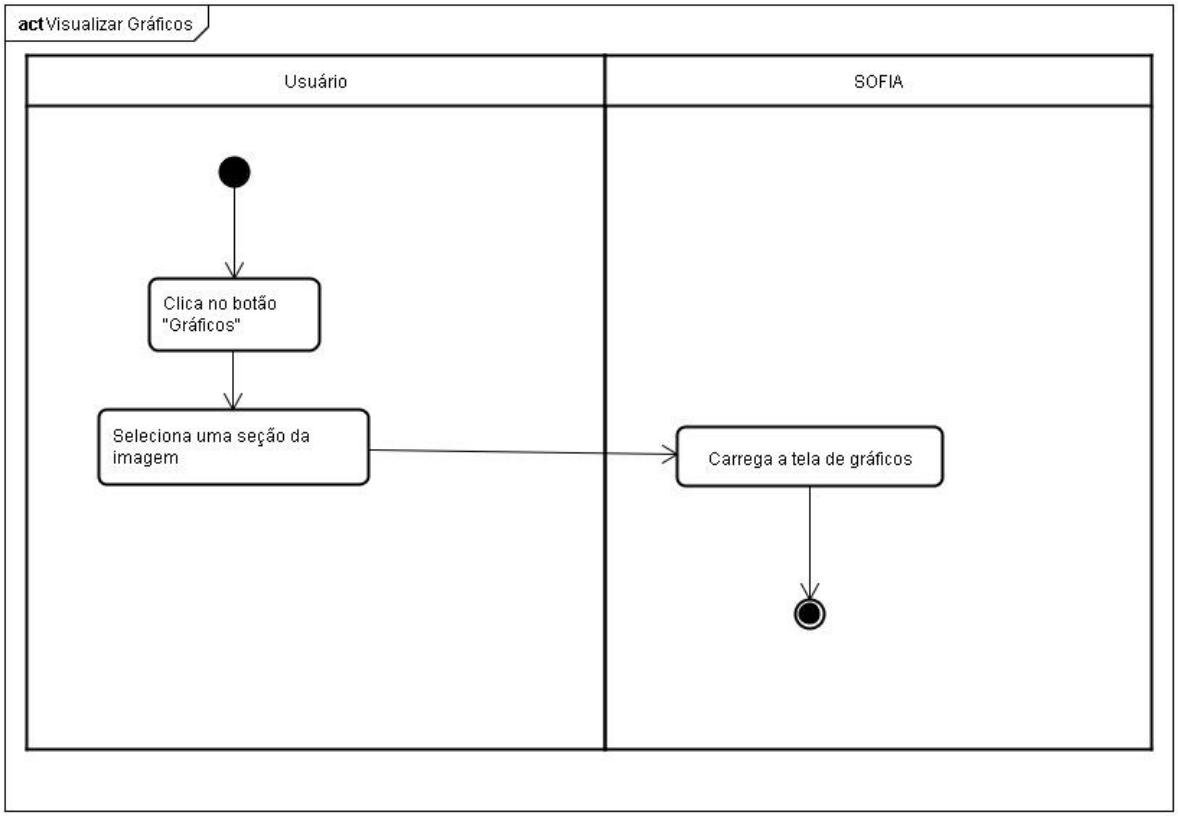


Figura 46 - Atividades - Visualizar gráficos
Fonte – Trabalho de Conclusão de Curso

APÊNDICE 8 – DIAGRAMA DE PACOTES

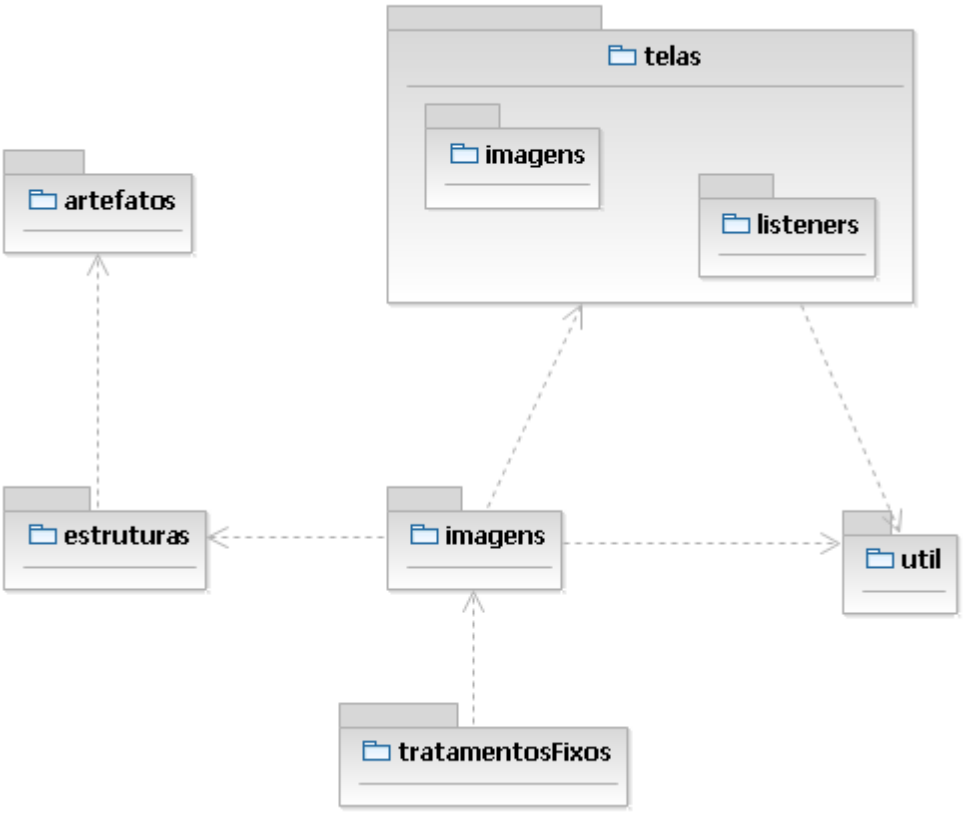


Figura 47 - Diagrama de pacotes
Fonte – Trabalho de Conclusão de Curso

APÊNDICE 9 – DIAGRAMA DE COMPONENTES

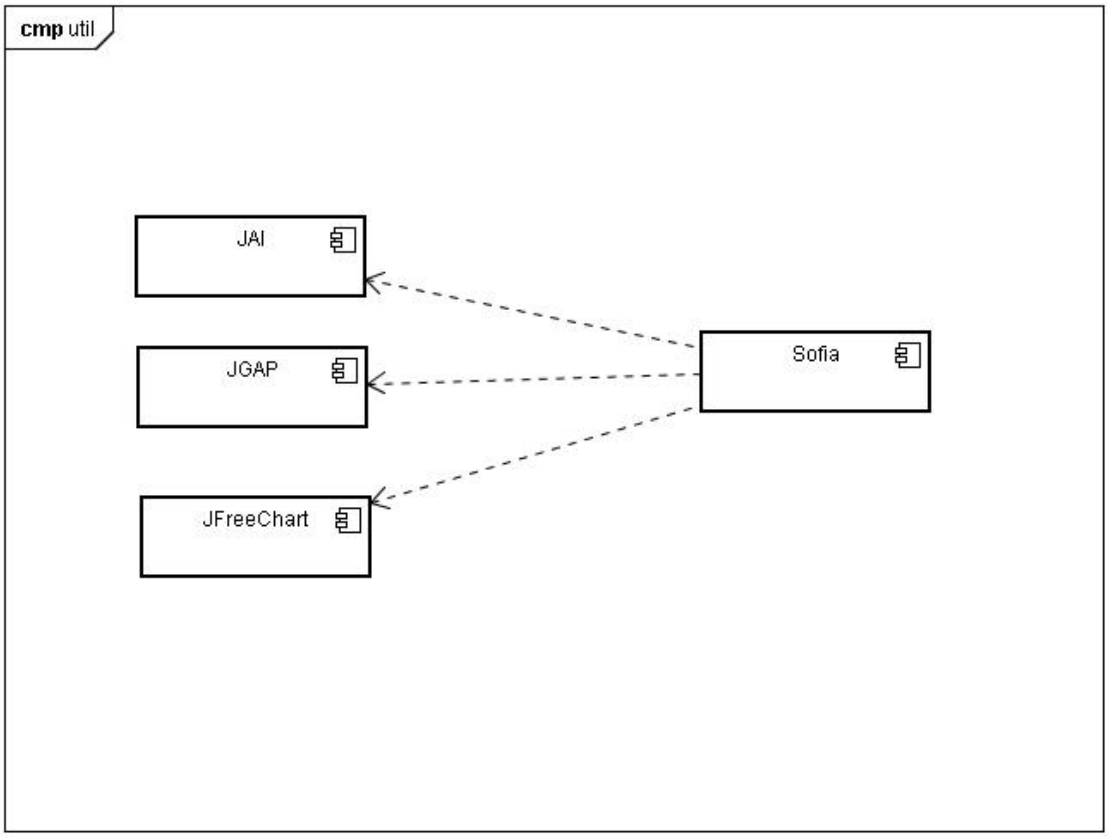


Figura 48 - Diagrama de componentes
Fonte – Trabalho de Conclusão de Curso

APÊNDICE 10 – RESULTADO EVINCI

174A AUTOMAÇÃO DO PROCESSO DE INTERPRETAÇÃO DE GÉIS 2D DE ELETROFORESE UTILIZANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL

Aluno de Iniciação Científica: Vanely de Souza (Bolsista) (PIBIC - CNPq)
Nº de Registro do Projeto de Pesquisa no BANPESQ/THALES: 2008022512
Orientador: Roberto Tadeu Raittz, Dr.
Colaboradores: Luciano Huergo, Michelly Alves Coutinho Gehlen, Juliana Helena Tibães, Aline Ariane Schultz, Angélica Inajá Juliani e Letícia Fernandes de Jesus.
Departamento: - Setor: Educação Profissional e Tecnológica
Palavras-chave: imagens digitais de géis 2D, algoritmo genético, inteligência artificial.
Área de Conhecimento: 1.03.04.00-2

O projeto desenvolvido visa à construção de um software capaz de efetuar o pré-processamento das imagens digitais de géis 2-D resultantes das pesquisas proteômicas. Utiliza técnicas de inteligência artificial para a implantação de conceitos de Visão Computacional. A meta é diminuir os ruídos presentes nas imagens resultantes da corrida do gel, facilitando a detecção dos spots pelas soluções computacionais existentes. O modelo faz uso da linguagem de programação JAVA e suas bibliotecas padrão. Com uso de Algoritmo Genético, técnica de inteligência artificial a imagem poderá ser processada de maneira "inteligente" o que permite detectar os spots e eliminar os ruídos presentes. O teste da aplicação será realizado através de comparações dos resultados apresentados pelo software utilizado pelos pesquisadores da UFPR. Essa comparação é estabelecida com imagens submetidas e não submetidas ao pré-processamento proposto pela ferramenta desenvolvida nesse projeto.

UFPR
UNIVERSIDADE FEDERAL DO PARANÁ

17° EVINCI
Evento de Iniciação Científica

Universidade Federal do Paraná
Pró-Reitoria de Pesquisa e Pós-Graduação
Coordenadoria de Iniciação Científica e Integração Acadêmica

CERTIFICADO

Certificamos que o(a) acadêmico(a)

VANELY DE SOUZA

Participou do 17° Evento de Iniciação Científica (17° EVINCI) da Universidade Federal do Paraná, realizado em Curitiba – PR no período de 21 a 23 de Outubro de 2009, onde apresentou o trabalho intitulado

AUTOMAÇÃO DO PROCESSO DE INTERPRETAÇÃO DE GEIS 2D DE ELETROFORESE UTILIZANDO TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL

Curitiba, 23 de Outubro de 2009.

Profª Drª Claudia Pereira Krueger
Coordenadora de Iniciação Científica
e Integração Acadêmica

Prof. Dr. Sérgio Scheer
Pró-Reitor de Pesquisa e Pós-Graduação

Franciele Cristine Pereira	155	8,36	
Francielle da Silva Borges	156	8,7	
Gabriel Kaetan Baio Ferreira	157	9,6	
Guilherme Purcote dos Santos	158	8,4	
Guilherme Taborda Ribas	159	8	
Gustavo Stentzler Garcia de Lima	160	8,3	
Henrique de Oliveira Okada	161	9,1	
Joao Paulo de Oliveira Souza	162	9,8	1° Lugar- Química III
Juliana Kovaleski de Oliveira	163	8,5	
Juliana Mari Jo	164	9,1	
KARINE PRISCILA NAIDEK	165	8,77	
Kassia dos Santos	166	9,3	
Leandro Blachechen	167	8,3	
Ligia Marilia Piai Almeida	168	9,5	
Luiz Antonio Bortolli Junior	169	8,8	
Marcus Vinicius Brandalize	170	9,7	1° Lugar- Química V
MARIA HELENA VERDAN	171	9,5	
Otavio Fuganti	172	8,1	
Patrícia Arianne Cornelsen	173	8,76	
Ricardo Ferraz da Silva	174	9	
Vanely de Souza	174A	9,45	
Camilo Dallagnol	175	8,46	
Gustavo Lenci Marques	176	8,83	
Anderson Domingues Gomes	177	7,4	*Ausência da apresentação oral justificada
Andressa Glinski	178	8,8	
Bruna Andrade Pereira	179	8,2	
Camila Valente Maiolino	180	10	1° Lugar- Biologia Celular III
Caroline Grisbach	181	8,8	
Danieli Dietrich Moura Costa	182	9,1	1° Lugar- Biologia Celular I
Debora Cristina Cestaro	183	9,33	
Ediely Layana Oliveira Coletto	184	8,5	
Fernanda Gatto de Almeida	185	9,1	
Fernanda Nunes Souza	186	8,5	
Flavia Yoshie Yamamoto	187	9,6	
Gabriel Otto Meissner	188	8,8	
Gustavo Rodrigues Rossi	189	8,4	
Isabela Castro Rossato	190	8,7	
IZABELA PAULINI DE JESUS	191	8,7	
Lucas Ferrari de Andrade	192	8,7	
Luis Paulo Silveira Alves	193	8,8	
Maria Rosa Dmengeon Pedreiro	194	8,5	
Nadia Sabchuk	195	7,3	*Ausência da apresentação oral justificada
Priscila Krebsbach	196	7,5	*Ausência da apresentação oral justificada
Rafael José Munhoz de Oliveira	197	9,43	1° Lugar- Biologia Celular II
Renata Rank de Miranda	198	9,03	
Adriano Alves Stefanello	199	9,5	
Alessandra Amélia Pirana	200	8,8	
Alex Vinicius Lopes Machado	201	8,9	
Aline Braun	202	9,4	1° Lugar- Bioquímica IV
Ana Carolina Origa Alves	203	8,7	
Andréia Aparecida Beraldo	204	9,24	
Andre Luiz Lopes Martins	205	9,3	